Subset Selection Using Nonlinear Optimization

A Thesis

Submitted to the Faculty

of

Drexel University

by

Trip Denton

in partial fulfillment of the

requirements for the degree

of

Doctor of Philosophy in Computer Science

2007

Dedications

To world peace

Acknowledgements

This research would not have been possible without the support and encouragement of many individuals. Without the advice and encouragement of my dear friend and advisor, Dr. Ali Shokoufandeh, I might never have pursued an advanced degree. I would also like to thank Dr. Jeremy Johnson who was instrumental in my decision to study at Drexel.

The support of my family has been invaluable, their constant encouragement provided inspiration and support at all times. I would like to thank my parents for their support and encouragement. I am indebted to my wife, for her patience and thoughtful suggestions. I would also like thank my son for his patience.

My research would not have been possible without the work of Jeff Abrahamson, Lars Bretzner, M. Fatih Demirci, Sven Dickinson, Luc Florack, Connie Gomez, Ahmed E. Hassan, Frans Kanters, Jay Kothari, Spiros Mancoridis, Matthew Maycock, Ko Nishino, John Novatnack, Maher Salah, Ali Shokoufandeh, Wei Sun, Filippos I. Vokolos, and Xun Zhou.

I would also like to thank Peter Bogunovich, Bruce Char, Leeann Crowers, Pawel Hitczenko, Gaylord Holder, Louis Kratz, Yelena Kushleyeva, Walt Mankowski, Andrea Negro, Thu Nguyen, Adam J. O'Donnell, Jeffrey Popyack, Vassilis Prevelakis, Craig Schroeder, and Servesh Tiwari.

Finally, I would like to thank the members of my thesis committee, Dr. Ali Shokoufandeh, Dr. Spiros Mancoridis, Dr. Ko Nishino, Dr. Dario Salvucci, and Dr. Jianbo Shi for their help in making this work possible.

—

Funded in part by grants form National Science Foundation (NSF/EIA 02-05178) and Office of Naval Research (ONR-N000140410363).

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation or the other supporting government and corporate organizations.

Table of Contents

List of Tables

List of Figures

Abstract

Subset Selection Using Nonlinear Optimization

Trip Denton
Advisor: Ali Shokoufandeh, Ph.D.

A common problem in computer science is how to represent a large dataset in a smaller more compact form. This thesis describes a generalized framework for selecting canonical subsets of data points that are highly representative of the original larger dataset. The contributions of the work are formulation of the subset selection problem as an optimization problem, an analysis of the complexity of the problem, the development of approximation algorithms to compute canonical subsets, and a demonstration of the utility of the algorithms in several problem domains.

1. Introduction and Background

1.1 Introduction

Many problems in computer science share a common theme, a large dataset must somehow be represented in a smaller more compact form. The motivations for this vary by application and problem domain as do the methods for representation. Perhaps the simplest way to represent a large dataset is to count the data points, and for some applications this may be sufficient, *i.e.* 4027 neutrons were detected in a one second interval.

Sometimes the dataset needs to be compressed for transmission bandwidth or storage reasons, and the compact form must retain enough information about the original dataset so that it can be reconstituted. Such compression algorithms frequently have requirements that they recreate the original dataset with little or no error.

In other cases, the need for seeking a more compact form is motivated by the efficiency of algorithms that must process the data. For example, in the case of computer vision applications, digital images may be represented by visual features detected in the images. In this representation, each feature is a data point. The algorithms that are used to extract meaning from these feature datasets are often highly dependent on the size of the input set. In many cases the complexity of the algorithms induces severe restraints on the size of the input set that can be efficiently processed.

To alleviate this problem, computer scientists are often forced to choose between subsampling the underlying dataset or somehow tweaking the data acquisition process in order limit the amount of data collected. Often there are domain specific methods of transforming the data and representing it in some abstracted form that is reduced in size. In the case of computer vision, this transformation is known as feature detection, which is an integral step in many computer vision tasks. Features detected in an image of an object form an abstract representation of the object which can be used by higher level vision processes such as localization and recognition.

The problem of representing a large dataset with a smaller more compact form can be accomplished though the following methods.

- New data points can be synthesized to represent clusters of the original data points.
- A subset of the original data points can be selected to represent the data set.

Liu and Motoda [71] refer to these methods as *feature transformation* and *subset selection*, respectively. The focus of this thesis is on the development of a new method of subset selection which is not domain specific. This method can be used to select a canonical subset of data points that are highly representative of the original larger dataset.

The development of the canonical subset method was motivated by the fact that in some cases feature transformation may not be practical, relevant, or even possible. For example, again consider the task of object recognition. Objects may be represented by hundreds of features and feature transformation will construct new features where none existed, misrepresenting the object.

Further motivation comes from the apparent drawbacks of current methods of subset selection. Methods based on finding elements close to centroids are often susceptible to noise and outliers. If there are multiple elements equidistant from the centroid it is unclear which is the best choice. These methods do not take the global topology of the original set into account, they are based on greedy local choices. Alternative methods based on random sampling may miss small structures completely. The goal is to expose the underlying structure of the data and have the global topology drive the subset selection process.

The contributions of this thesis are as follows:

1. A new characterization of subset selection as an optimization problem.

2. The development of an algorithmic framework for selecting canonical subsets based on similarity measures.
3. Examples of applications in problem domains where the canonical subset algorithms have been shown to be useful.

1.2 Background

Data compression is similar to subset selection in that it seeks to represent an input signal or set of data points as efficiently (in terms of space) as possible. In 1948 Claude Shannon [101] described how to measure the information content in a series of n events whose probabilities are $(p_1, p_2, \ldots, p_n)$. He defined the entropy H as

$$H = -K \sum_{i=1}^{n} p_i \log p_i,$$

where K is constant. By describing how information content could be measured, Shannon provided a way to calculate the theoretical limit on how much a signal can be compressed.

A classic method of data compression is Huffman coding [10]. Huffman codes are a method of data compression where an input stream of characters (patterns) is encoded using variable length codewords. These codewords are *prefix* codes, meaning that no two codewords have a common prefix. The algorithm for constructing the codes looks at the frequency of the characters and assigns codewords that are short to characters that have a high frequency and codewords that are long to characters with low frequency. The codewords that are used in this method are related to the input characters by frequency only and as such are arbitrary representations of them. Data compression methods such as Huffman codes preserve storage space, reduce transmission time, and facilitate error correction. With some exceptions [100, 106], high level algorithms are formulated to work with uncompressed data. So in general, the data must be decompressed, making any computational gains obtained from working with a smaller input set (the compressed data) somewhat irrelevant for higher level algorithms that might be of interest.

Rate distortion theory as described by Cover [11], represents an input sequence, X, as a set of reproduction points, $\hat{X}$, defined by a mapping $f : X \rightarrow \hat{X}$. The *distortion*, d, between two sequences, X and $\hat{X}$ is defined as

$$d(X, \hat{X}) = \frac{1}{n} \sum_{i=1}^{n} \delta(x_i, f(x_i)),$$

where $n = |X|$ and $\delta(x_i, f(x_i))$ is some distortion function. Commonly used distortion functions are the Hamming distortion, *i.e.* $\delta(x, y) = 0$ if $x = y$ and 1 otherwise, and the squared error distortion, $\delta(x, y) = (x - y)^2$. The goal for a given $k,\ k < |X|$, is to find a mapping, $f : X \rightarrow \hat{X}$, if one exists, such that the distortion is minimized. This method does not use a subset of X as $\hat{X}$, so it differs from the canonical set method.

Also related to distortion rate theory is the information bottleneck method of Tishby *et al.* [110]. In their method an additional source of relevant information Y, is available which is used with X and $\hat{X}$ as described in rate distortion theory. This in conjunction with β, a tuning parameter, allows the method to produce a desired trade-off between compressed size and fully detailed quantization.

Gordon *et al.* [33] used the information bottleneck method for unsupervised clustering of image databases. They applied the method to discrete image histograms and probabilistic continuous feature sets of images based on Gaussian mixture models. In their work, they compare their method with that of histogram intersection, indicating that methods based on the information bottleneck give superior results. The information bottleneck method differs from the canonical set method in that no assumption is made about the availability of an additional source of relevant information.

The clustering problem is also related to the subset selection problem. In clustering, some measure of similarity between data points in the input dataset is defined. The dataset is then partitioned such that data points in any one partition are more similar to other data points in the same partition than to data points in other partitions. The resulting partitions can then be approximated by centroids or other synthetic data points. Typically clustering methods require as an input the desired number of clusters, k.

The most widely used clustering algorithm, K-means [78], clusters n data points into k clusters. The algorithm begins by creating k random cluster centroids, and data points are assigned to the nearest centroid. Then the centroids of the clusters are recalculated. The data points are reassigned to the nearest centroid; and this process is repeated until the assignment of data points to clusters stabilize. There are many variations to the K-means algorithm, yet all suffer from the same drawbacks. The centroids are synthetic data points and a nearest neighbor search must be done to find a representative element that is closest to the centroid. K-Means is also sensitive to outliers, more importantly, the clusters are produced as the result of a greedy algorithm that bases its results on local decisions rather than the global topology of the system. Incorporating global constraints such as maximizing the distance between centroids or minimizing the distance to the nearest centroid can be difficult to encode.

A classic problem in computer vision related to clustering is image segmentation. "Segmentation is the process of partitioning an image into disjoint and homogeneous regions" [77]. Intuitively, segmentation decomposes an image into groups of similar pixels. Shi and Malik [102] formulated image segmentation as a graph partitioning problem through the use of normalized cuts. The *normalized cut* ($Ncut$) is defined as

$$Ncut(A,B) = \frac{cut(A,B)}{asso(A,V)} + \frac{cut(A,B)}{asso(B,V}, $$

where $asso(A,V) = \sum_{u\in A, t\in V} w(u,t)$ is the total connection from nodes A to all nodes in the graph and $cut(A,B) = \sum_{u\in A, v\in B} w(u,v)$, the sum of the weights of the cut edges. They then used the spectral properties of distance matrices to perform the clustering. Specifically, they looked at the second smallest eigenvector as an indicator for a binary split. The process is run recursively, each time selecting a group of similar data points (pixels). This is in contrast to the canonical set method which directly selects representative data points. The representative data points in a canonical set are in fact highly dissimilar since they must represent the entire dataset.

The problem of feature or attribute selection has also been studied in the context of the dimensionality curse and dimensionality reduction [5, 4, 90]. This is mainly motivated by that fact that in large semi-structured databases there are many attributes which are correlated with the others, and the need for feature reduction and attribute selection is motivated by the fact that many datasets can largely be well-approximated in fewer dimensions.

Liu *et al.* [71] provide a broad overview of the feature transformation and subset selection methods. Liu *et al.* [72] describe the filter versus wrapper model for feature selection in the context of inductive learning. Koller and Sahami [57] describe a method for feature subset selection based on information theory. Langley [63] formulates the problem as search problem and categorizes a number of methods according to search strategy. The feature selection discussed in these papers refers to the selection of which attributes of the data points have the most predictive power, *i.e.* which are most useful for classifying new data points. These attributes are selected using a set of *training* data points where the classification is known. This differs from the canonical set method where a representative subset is selected without any prior knowledge of the dataset (*e.g.* a training set).

More related to the work presented here is that of Sun *et al.* [108]. They explored the use of a genetic algorithm for the purpose of selecting a subset of features from a training set and used this subset for the object recognition tasks of vehicle detection and face recognition. The canonical set method work differs from their approach in that no training set is required.

The canonical set method is also related to the recent work on the identification of canonical views of 3D objects [76, 89]. The objective is to select a subset of object views that best represents an object. These canonical views can then be used in technical drawings and computer visualizations. Canonical views are similar to the prototype views described by Cyr and Kimia [13], which were used for 3D object recognition.

The approach here is to model subset selection as an optimization problem. The canonical set method was inspired by the success of Goemans and Williamson [30] on the MAX-CUT problem in graphs, which used semidefinite programming (SDP) relaxations to obtain an improved approximation algorithm. SDP relaxations for problems related to MAX-CUT such as MAX-BISECTION and MAX-CUT with given sizes of cuts have also been studied by others [23, 28, 34, 116]. See Goemans [29] and Mahajan and Ramesh [79] for a survey of recent results and applications of SDP. Recently, SDP techniques have found an increasing number of applications in computer vision tasks such as graph matching [2, 99], segmentation [52], and shape from shading [117].

1.3 Overview of the Thesis

The rest of this thesis is organized as follows, Chapter 2 presents the formulation of the canonical set problem. It explains how the problem can be represented as an optimization on appropriately defined graph and presents quadratic integer programming formulations for three different types of canonical sets. To motivate the development of approximation algorithms, Chapter 3 presents proofs of intractability and describes algorithms to compute approximate solutions to the problems described in Chapter 2.

Chapter 4 presents experiments that examine the question of how sensitive canonical sets are to outliers and shows canonical sets of some regular structures. Chapter 5 presents results evaluating the utility of the canonical set method of subset selection on the computer vision problem of 2D view selection. Chapter 6 presents an extensive set of experiments where canonical subsets of image features are used to localize objects under occlusion.

Chapter 7 shows how the canonical set method can be used to select subsets of image features for the purpose of image reconstruction. Although the canonical set algorithms were originally designed for computer vision applications, they are general in nature. As proof of their generality, Chapter 8 presents results that show the canonical set method is also useful in the field of software engineering. Lastly, Chapter 9 presents conclusions as well as a discussion of future work.

2. Problem Formulation

2.1 Introduction

This chapter presents the formulation of the canonical set problem. In Section 2.2, some notation is introduced and it is shown how the canonical set problem can be represented as an optimization on appropriately defined graph. Section 2.3 introduces the properties of canonicals sets and Section 2.4 introduces their inverses. In Section 2.5, formulations for canonical set constraints are presented. Section 2.6 introduces the concept of Pareto optimality and explains how multiple objectives can be combined. Finally, Section 2.7 presents quadratic integer programming formulations for three different types of canonical sets.

2.2 Graph Representation

Let $\mathcal{P} = \{p_1, ..., p_n\}$ be a set of n data points, and let $\mathcal{P}^*$ denote the canonical subset of $\mathcal{P}$. Let $\mathcal{S} : \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{R}^{\geq 0}$ be a similarity function where $\mathcal{S}_{ij}$ is the similarity between data points p_i and p_j. Assume the similarity measure is symmetric, *i.e.* $\mathcal{S}_{ij} = \mathcal{S}_{ji}$. Without loss of generality, assume that all similarities fall in the interval $[0, 1]$, with 0 meaning there is no similarity between the data points and 1 meaning the data points are indistinguishable. Define a matrix $\mathcal{W}$, where $\mathcal{W}_{i \neq j} = \mathcal{S}_{ij}$ and $\mathcal{W}_{ii} = 0$.

Construct an undirected edge weighted graph $G = G(\mathcal{P})$, where the data points are represented by vertices, and the edges between the vertices have weights corresponding to the measure of similarity between the vertices. If the similarity between a pair of data points is zero, no edge exists between the respective vertices. Let $\mathcal{A} \in \{0, 1\}^{n \times n}$ encode the adjacency matrix of graph G,

$$\mathcal{A}_{ij} = \begin{cases} 1 & \text{if } \mathcal{W}_{ij} > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Using the graph representation, (see Figure 2.1) it is easy to see that with respect to a given $\mathcal{P}^*$, all of the edges of the graph fall into three categories. An *intra* edge connects two vertices which are both in $\mathcal{P}^*$, a *cut* edge has exactly one vertex in $\mathcal{P}^*$, and an *extra* edge has exactly zero vertices in $\mathcal{P}^*$. This model can be extended to include vertex weights

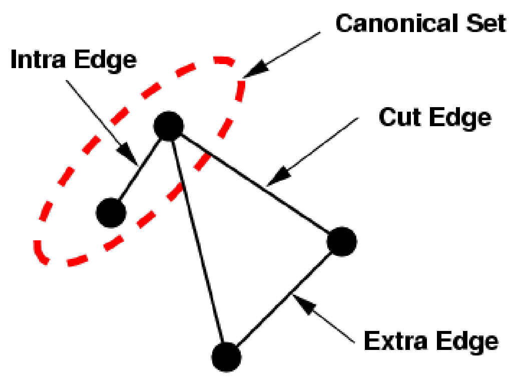


Figure 2.1: Representing the canonical set as a graph; the edges can be categorized as intra, cut, or extra.

by associating a scalar value with each data point, $\{t_1, ..., t_n\},\ t_i \in \mathbb{R}^+ \ \forall\ 1 \leq i \leq n$, that will notationally be called the *stability* of the data point.

To formulate the canonical set problem, a set indicator variable [29], $y_{n+1} \in \{-1, 1\}$ is introduced. Then for each data point, p_i, create an indicator variable

$$y_i = \begin{cases} y_{n+1} & \text{if } p_i \in \mathcal{P}^* \\ -y_{n+1} & \text{otherwise.} \end{cases}$$

The set indicator variable, y_{n+1}, acts as a reference for membership in the canonical set $\mathcal{P}^*$. In an optimal solution, feature p_i is a member of the canonical set only if $y_i = y_{n+1}$. Note that y_{n+1} can equal -1 or 1, and as a result

$$p_i \in \mathcal{P}^* \Leftrightarrow \frac{1 + y_i y_{n+1}}{2} = 1 \text{ and } p_i \notin \mathcal{P}^* \Leftrightarrow \frac{1 - y_i y_{n+1}}{2} = 1.$$

2.3 Properties of Canonical Sets

Using the notation described in Section 2.2, the cardinality of the canonical set $\mathcal{P}^*$ may be written as

$$|\mathcal{P}^*| \quad = \quad \frac{1}{2} \sum_{i=1}^{n} (1 + y_i y_{n+1}). \tag{2.1}$$

In a similar manner the stability of the canonical set $\mathcal{P}^*$ may be written as

$$\text{Stability}(\mathcal{P}^*) \quad = \quad \frac{1}{2} \sum_{i=1}^{n} t_i (1 + y_i y_{n+1}), \tag{2.2}$$

the sum of the weights of the cut edges may be written as

$$\text{Cut}(\mathcal{P}^*) \quad = \quad \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} (1 - y_i y_j), \tag{2.3}$$

the sum of the weights of the intra edges may be written as

$$\text{Intra}(\mathcal{P}^*) \quad = \quad \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} (1 + y_i y_{n+1})(1 + y_j y_{n+1}), \tag{2.4}$$

and the sum of the weights of the extra edges may be written as

$$\text{Extra}(\mathcal{P}^*) \quad = \quad \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} (1 - y_i y_{n+1})(1 - y_j y_{n+1}). \tag{2.5}$$

The properties of canonical sets in Equations (2.1), (2.2), (2.3), (2.4), and (2.5) are summarized in Table 2.1.

Property	**Formulation**	**Description**
Size($\mathcal{P}^*$)	$\frac{1}{2}\sum_{i=1}^{n}(1+y_iy_{n+1})$	Cardinality ($\lvert\mathcal{P}^*\rvert$) of canonical set
Stability($\mathcal{P}^*$)	$\frac{1}{2}\sum_{i=1}^{n}t_i(1+y_iy_{n+1})$	Stability of canonical set
Cut($\mathcal{P}^*$)	$\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-y_iy_j)$	Sum of cut edge weights
Intra($\mathcal{P}^*$)	$\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1+y_iy_{n+1})(1+y_jy_{n+1})$	Sum of intra edge weights
Extra($\mathcal{P}^*$)	$\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-y_iy_{n+1})(1-y_jy_{n+1})$	Sum of extra edge weights

Table 2.1: Properties of canonical sets

2.4 Property Inverses

To facilitate formulation of optimization problems, it is useful to note that each of the properties in Table 2.1 has an inverse. The inverse properties are designed such that if Q is a property, then an objective function minimizing Q is the same as maximizing Q^{-1} and vice versa. The formulations can be deduced by examining the inverse partitions of the graph G. For instance, the cardinality of the set $P \setminus P^*$ may be written as

$$\begin{aligned} |P \setminus P^*| &= \text{Size}^{-1}(\mathcal{P}^*) \\ &= n - \frac{1}{2}\sum_{i=1}^{n}(1+y_iy_{n+1}) \\ &= \frac{1}{2}\sum_{i=1}^{n}(1-y_iy_{n+1}). \end{aligned} \tag{2.6}$$

Likewise

$$\text{Stability}^{-1}(\mathcal{P}^*) = \frac{1}{2}\sum_{i=1}^{n} t_i(1 - y_i y_{n+1}), \tag{2.7}$$

and

$$\text{Cut}^{-1}(\mathcal{P}^*) = \frac{1}{4}\sum_{i,j} \mathcal{W}_{ij}(1 + y_i y_j). \tag{2.8}$$

Using the facts that $\mathcal{W}$ is symmetric and $y_{n+1}^2 = 1$,

$$\begin{aligned}
\text{Intra}^{-1}(\mathcal{P}^*) &= \text{Cut}(\mathcal{P}^*) + \text{Extra}(\mathcal{P}^*) \\
&= \frac{1}{4}\sum_{i,j} \mathcal{W}_{ij}(1 - y_i y_j) + \frac{1}{4}\sum_{i,j} \mathcal{W}_{ij}(1 - y_i y_{n+1})(1 - y_j y_{n+1}) \qquad (2.9) \\
&= \frac{1}{2}\sum_{i,j} \mathcal{W}_{ij} - \frac{1}{2}\sum_{i=1}^{n}\left(y_i y_{n+1} \sum_{j=1}^{n} \mathcal{W}_{ij} \right),
\end{aligned}$$

and

$$\begin{aligned}
\text{Extra}^{-1}(\mathcal{P}^*) &= \text{Cut}(\mathcal{P}^*) + \text{Intra}(\mathcal{P}^*) \\
&= \frac{1}{4}\sum_{i,j} \mathcal{W}_{ij}(1 - y_i y_j) + \frac{1}{4}\sum_{i,j} \mathcal{W}_{ij}(1 + y_i y_{n+1})(1 + y_j y_{n+1}) \qquad (2.10) \\
&= \frac{1}{2}\sum_{i,j} \mathcal{W}_{ij} + \frac{1}{2}\sum_{i=1}^{n}\left(y_i y_{n+1} \sum_{j=1}^{n} \mathcal{W}_{ij} \right).
\end{aligned}$$

Equations (2.6), (2.7), (2.8), (2.9), and (2.10) are summarized in Table 2.2.

Property	Formulation	Description
Size$^{-1}(\mathcal{P}^*)$	$\frac{1}{2}\sum_{i=1}^{n}(1-y_iy_{n+1})$	$\lvert P \setminus P^*\rvert$
Stability$^{-1}(\mathcal{P}^*)$	$\frac{1}{2}\sum_{i=1}^{n}t_i(1-y_iy_{n+1})$	Stability of $P \setminus P^*$
Cut$^{-1}(\mathcal{P}^*)$	$\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1+y_iy_j)$	Sum of uncut edge weights
Intra$^{-1}(\mathcal{P}^*)$	$\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-y_iy_j)$ $+\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-y_iy_{n+1})(1-y_jy_{n+1})$	Sum of non-intra edge weights
Extra$^{-1}(\mathcal{P}^*)$	$\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-y_iy_j)$ $+\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1+y_iy_{n+1})(1+y_jy_{n+1})$	Sum of non-extra edge weights

Table 2.2: Property inverse formulations

2.5 Constraints

One consequence of the notation described in Section 2.2, is that canonical sets have a required constraint,

$$y_i \in \{-1,+1\},\ 1 \leq i \leq n+1. \tag{2.11}$$

Suppose a lower bound, k_{min}, on the size of the canonical set is desired. This may be expressed as the constraint,

$$\frac{1}{2}\sum_{i=1}^{n}(1+y_iy_{n+1}) - k_{min} \geq 0. \tag{2.12}$$

In a similar fashion, an upper bound, k_{max}, may be expressed as

$$k_{max} - \frac{1}{2}\sum_{i=1}^{n}(1 + y_i y_{n+1}) \geq 0. \tag{2.13}$$

A set of vertices $V' \subseteq V$ is a *dominating set* for $G(V, E)$ if for every vertex $u \in V \setminus V'$, there exists a vertex $v \in V'$, with $(u, v) \in E$. A dominating set constraint may be written as

$$(1 + y_i y_{n+1}) + \sum_{j=1}^{n} \mathcal{A}_{ij}(1 + y_j y_{n+1}) \geq 2. \tag{2.14}$$

Constraints involving first order logic statements such as $\wedge, \vee$, and $\neg$, may be formulated in in a similar way.

2.6 Combining Objectives

If canonical subsets are constructed with multiple objective functions, the objectives may need to be combined in some manner. For an overview of the theory behind combining objective functions, the reader is referred to Miettinen [81]. The general formulation of such multi-objective optimization problems is as follows [22, 81]:

$$\begin{array}{rl} \text{Maximize} & \mathcal{F}(\mathcal{X}) = \{f_1(\mathcal{X}), f_2(\mathcal{X}), \ldots, f_m(\mathcal{X}))\} \\ \text{Subject to} & \mathcal{X} \in \Gamma, \end{array}$$

where $\mathcal{F}(\mathcal{X}) = \{f_1(\mathcal{X}), \ldots, f_m(\mathcal{X})\}$ is the set of objective functions and Γ is the feasible set.

In cases such as these, a trade-off optimality condition known as *Pareto* optimality may be used [22]. Specifically, a solution $\mathcal{X}^*$ is called Pareto optimal if there is no $\mathcal{X} \in \Gamma$ such that $\mathcal{F}(\mathcal{X}) \geq \mathcal{F}(\mathcal{X}^*)$; that is, $\mathcal{X}^*$ is lexicographically optimal compared to any sub-optimal solution $\mathcal{X}$. The set of all Pareto optimal solutions $\mathcal{X}^* \in \Gamma$ is denoted by Γ_{Par}, the *Pareto set*. The Pareto set can be thought of as the boundary of the image of the feasible set (see Figure 2.2), and a Pareto optimal solution is a point on the boundary.

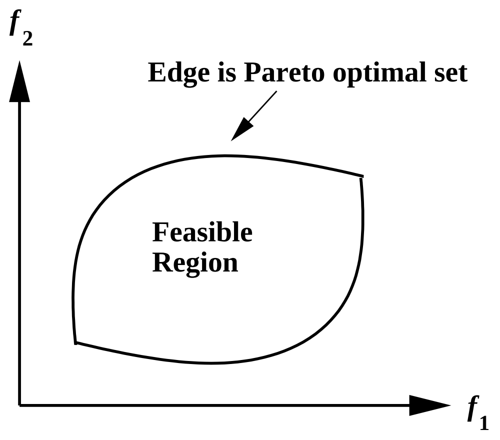


Figure 2.2: Pareto Set.

If in a multi-objective optimization problem, the functions $f_i(\mathcal{X})$, $1 \leq i \leq m$, are all convex functions, then the optimal solution $\mathcal{X}^*$ of the following single-objective problem belongs to the Pareto set of problem $\mathcal{F}(\mathcal{X})$ [81]:

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{m} \lambda_i f_i(\mathcal{X}) \\ \text{Subject to} \quad & \mathcal{X} \in \Gamma, \\ & \lambda_i \geq 0 \quad \forall\, 1 \leq i \leq m, \\ & \sum_{i=1}^{m} \lambda_i = 1. \end{aligned}$$

This weighting of objectives finds a Pareto optimal point for a given set of λ values. Because of the implicit trade-off obtained through the weighting, comparison between solutions obtained with a different set of λ values might not be possible. For example, if the

Cut($\mathcal{P}^*$) and Intra($\mathcal{P}^*$) are to be minimized, the λ values will prescribe a specific importance to the relationship between the sum of the cut edges and the sum of the intra edges. Finally note that all of the functions listed in Tables 2.1 and 2.2 are convex since their Hessians are non-negative semidefinite.

2.7 Canonical Set Problem Formulation

Using the machinery developed in Sections 2.2-2.4 formulations of canonical subsets of $\mathcal{P}$ can be created. The formulations are presented as quadratic integer programming problems.

2.7.1 Minimum Dominating Maximum Cut (MDMC) Canonical Subset

Suppose a highly representative subset of some dataset is desired, and there is a requirement for the subset to be as small as possible. In addition, some of the pairs of data points are completely dissimilar. The goal is to identify a canonical subset, $\mathcal{P}^*$ with the following attributes:

1. Data points in $\mathcal{P}^*$ are maximally similar to data points in $P \setminus P^*$.

2. The size of the canonical set is as small as possible.

3. Every data point is either in $\mathcal{P}^*$ or is similar to a data point in $\mathcal{P}^*$.

A canonical subset with these attributes may be more formally described as the minimum dominating set with maximum cut-weight. Intuitively, such a set is a simultaneous solution to the minimum dominating set and the maximum cut problems in graph G. Using the notation described in Section 2.3 and 2.5, this canonical subset problem can be formulated as

$$\begin{aligned}
\text{Minimize}\quad & \text{Size}(\mathcal{P}^*),\\
\text{Maximize}\quad & \text{Cut}(\mathcal{P}^*),\\
\text{Subject to}\quad & (1 + y_i y_{n+1}) + \sum_{j=1}^{n} \mathcal{A}_{ij}(1 + y_j y_{n+1}) \geq 2,\ \forall\, 1 \leq i \leq n,\\
& y_i \in \{-1, +1\},\ \forall\, 1 \leq i \leq n+1.
\end{aligned}$$

Using the inverse properties given in Table 2.2, the problem formulation can be made consistent so that the objectives are all stated in maximization form,

$$\begin{aligned}
\text{Maximize}\quad & \text{Size}^{-1}(\mathcal{P}^*),\\
\text{Maximize}\quad & \text{Cut}(\mathcal{P}^*),\\
\text{Subject to}\quad & (1 + y_i y_{n+1}) + \sum_{j=1}^{n} \mathcal{A}_{ij}(1 + y_j y_{n+1}) \geq 2,\ \forall\, 1 \leq i \leq n,\\
& y_i \in \{-1, +1\},\ \forall\, 1 \leq i \leq n+1.
\end{aligned}$$

Substituting formulations from Tables 2.1 and 2.2 gives

$$\begin{aligned}
\text{Maximize}\quad & \frac{1}{2}\sum_{i=1}^{n}(1 - y_i y_{n+1}),\\
\text{Maximize}\quad & \frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1 - y_i y_j),\\
\text{Subject to}\quad & (1 + y_i y_{n+1}) + \sum_{j=1}^{n} \mathcal{A}_{ij}(1 + y_j y_{n+1}) \geq 2,\ \forall\, 1 \leq i \leq n,\\
& y_i \in \{-1, +1\},\ \forall\, 1 \leq i \leq n+1.
\end{aligned}$$

Combining the objective functions using the technique in Section 2.6 gives:

(MDMC):

$$\begin{array}{ll} \text{Maximize} & \lambda_1 \left(\frac{1}{2} \sum_{i=1}^{n} (1 - y_i y_{n+1}) \right) + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} (1 - y_i y_j) \right), \\ \text{Subject to} & (1 + y_i y_{n+1}) + \sum_{j=1}^{n} \mathcal{A}_{ij} (1 + y_j y_{n+1}) \geq 2, \ \forall\, 1 \leq i \leq n, \\ & y_i \in \{-1, +1\}, \ \forall\, 1 \leq i \leq n+1, \end{array}$$

where λ_1 and λ_2 are non-negative and $\lambda_1 + \lambda_2 = 1$.

2.7.2 Bounded Canonical Subset (BCS)

Suppose a highly representative subset of some dataset is desired, and there is a requirement for the members of the subset to be as dissimilar as possible. In addition, there is a need to be able to control the size of the subset. The goal is to identify a canonical subset, $\mathcal{P}^*$ with the following attributes:

1. Data points in $\mathcal{P}^*$ are minimally similar.

2. Data points in $\mathcal{P}^*$ are maximally similar to data points in $P \setminus P^*$.

3. The size of the canonical set is as least k_{min} and at most k_{max}.

Using the notation described in Section 2.3 and 2.5, this canonical subset problem can be formulated as

$$\begin{aligned}
\text{Minimize} \quad & \text{Intra}(\mathcal{P}^*), \\
\text{Maximize} \quad & \text{Cut}(\mathcal{P}^*), \\
\text{Subject to} \quad & \frac{1}{2}\sum_{i=1}^{n}(1 + y_i y_{n+1}) - k_{min} \geq 0, \\
& k_{max} - \frac{1}{2}\sum_{i=1}^{n}(1 + y_i y_{n+1}) \geq 0, \\
& y_i \in \{-1, +1\}, \ \forall\, 1 \leq i \leq n+1.
\end{aligned}$$

Using the inverse properties given in Table 2.2, the problem formulation can be made consistent so that the objectives are all stated in maximization form,

$$\begin{aligned}
\text{Maximize} \quad & \text{Intra}^{-1}(\mathcal{P}^*), \\
\text{Maximize} \quad & \text{Cut}(\mathcal{P}^*), \\
\text{Subject to} \quad & \frac{1}{2}\sum_{i=1}^{n}(1 + y_i y_{n+1}) - k_{min} \geq 0, \\
& k_{max} - \frac{1}{2}\sum_{i=1}^{n}(1 + y_i y_{n+1}) \geq 0, \\
& y_i \in \{-1, +1\}, \ \forall\, 1 \leq i \leq n+1.
\end{aligned}$$

Substituting formulations from Tables 2.1 and 2.2 gives

$$\begin{aligned}
\text{Maximize} \quad & \frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-y_iy_j)+\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-y_iy_{n+1})(1-y_jy_{n+1}),\\
\text{Maximize} \quad & \frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-y_iy_j),\\
\text{Subject to} \quad & \frac{1}{2}\sum_{i=1}^{n}(1+y_iy_{n+1})-k_{min}\geq 0,\\
& k_{max}-\frac{1}{2}\sum_{i=1}^{n}(1+y_iy_{n+1})\geq 0,\\
& y_i\in\{-1,+1\},\ \forall\, 1\leq i\leq n+1.
\end{aligned}$$

Combining the objective functions using the technique in Section 2.6 gives:

(BCS):

$$\begin{aligned}
\text{Maximize} \quad & \lambda_1\left(\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-y_iy_j)+\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-y_iy_{n+1})(1-y_jy_{n+1})\right)\\
& +\lambda_2\left(\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-y_iy_j)\right),\\
\text{Subject to} \quad & \frac{1}{2}\sum_{i=1}^{n}(1+y_iy_{n+1})-k_{min}\geq 0,\\
& k_{max}-\frac{1}{2}\sum_{i=1}^{n}(1+y_iy_{n+1})\geq 0,\\
& y_i\in\{-1,+1\},\ \forall\, 1\leq i\leq n+1,
\end{aligned}$$

where λ_1 and λ_2 are non-negative and $\lambda_1+\lambda_2=1$.

2.7.3 Stable Bounded Canonical Subset (SBCS)

Suppose a highly representative subset of some dataset is desired, and there is a requirement for the members of the subset to be as dissimilar as possible and as stable as possible. In addition, there is a need to be able to control the size of the subset. The goal is to identify a canonical subset, $\mathcal{P}^*$ with the following attributes:

1. Data points in $\mathcal{P}^*$ are minimally similar.

2. Data points in $\mathcal{P}^*$ are maximally similar to data points in $P \setminus P^*$.

3. Data points in $\mathcal{P}^*$ are maximally stable.

4. The size of the canonical set is as least k_{min} and at most k_{max}.

Using the notation described in Section 2.3 and 2.5, this canonical subset problem can be formulated as

$$\begin{array}{rl}
\text{Minimize} & \text{Intra}(\mathcal{P}^*), \\
\text{Maximize} & \text{Cut}(\mathcal{P}^*), \\
\text{Maximize} & \text{Stability}(\mathcal{P}^*), \\
\text{Subject to} & \displaystyle\frac{1}{2}\sum_{i=1}^{n}(1 + y_i y_{n+1}) - k_{min} \geq 0, \\
& \displaystyle k_{max} - \frac{1}{2}\sum_{i=1}^{n}(1 + y_i y_{n+1}) \geq 0, \\
& y_i \in \{-1, +1\}, \ \forall\, 1 \leq i \leq n+1.
\end{array}$$

Using the inverse properties given in Table 2.2, the problem formulation can be made consistent so that the objectives are all stated in maximization form,

$$\begin{aligned}
\text{Maximize} \quad & \text{Intra}^{-1}(\mathcal{P}^*), \\
\text{Maximize} \quad & \text{Cut}(\mathcal{P}^*), \\
\text{Maximize} \quad & \text{Stability}(\mathcal{P}^*), \\
\text{Subject to} \quad & \frac{1}{2}\sum_{i=1}^{n}(1 + y_i y_{n+1}) - k_{min} \geq 0, \\
& k_{max} - \frac{1}{2}\sum_{i=1}^{n}(1 + y_i y_{n+1}) \geq 0, \\
& y_i \in \{-1, +1\}, \ \forall\, 1 \leq i \leq n+1.
\end{aligned}$$

Substituting formulations from Tables 2.1 and 2.2 gives

$$\begin{aligned}
\text{Maximize} \quad & \frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1 - y_i y_j) + \frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1 - y_i y_{n+1})(1 - y_j y_{n+1}), \\
\text{Maximize} \quad & \frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1 - y_i y_j), \\
\text{Maximize} \quad & \frac{1}{2}\sum_{i=1}^{n} t_i(1 + y_i y_{n+1}), \\
\text{Subject to} \quad & \frac{1}{2}\sum_{i=1}^{n}(1 + y_i y_{n+1}) - k_{min} \geq 0, \\
& k_{max} - \frac{1}{2}\sum_{i=1}^{n}(1 + y_i y_{n+1}) \geq 0, \\
& y_i \in \{-1, +1\}, \ \forall\, 1 \leq i \leq n+1.
\end{aligned}$$

Combining the objective functions using the technique in Section 2.6 gives:

(SBCS):

$$\begin{aligned}
\text{Maximize} \quad & \lambda_1 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - y_i y_j) + \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - y_i y_{n+1})(1 - y_j y_{n+1}) \right) \\
& + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - y_i y_j) \right) + \lambda_3 \left(\frac{1}{2} \sum_{i=1}^{n} t_i (1 + y_i y_{n+1}) \right), \\
\text{Subject to} \quad & \frac{1}{2} \sum_{i=1}^{n} (1 + y_i y_{n+1}) - k_{min} \geq 0, \\
& k_{max} - \frac{1}{2} \sum_{i=1}^{n} (1 + y_i y_{n+1}) \geq 0, \\
& y_i \in \{-1, +1\}, \ \forall\, 1 \leq i \leq n+1,
\end{aligned}$$

where λ_1, λ_2, and λ_3 are non-negative and $\sum_{m=1}^{3} \lambda_m = 1$.

2.8 Conclusion

This chapter introduced notation and methods necessary for the formulation of canonical sets. By representing subset selection as an optimization on an appropriately defined graph, it was shown that various properties of the graph could be defined. These properties as well as relevant constraints were then formulated as quadratic integer equations. The notion of Pareto optimization, a method for combining objective functions was then discussed. Finally, three flavors of canonical subsets based on different combinations of the defined properties and constraints were presented.

3. Approximation and Proof of NP Hardness

3.1 Introduction

This chapter presents algorithms to compute approximate solutions to the problems described in Chapter 2. First in Section 3.2, it is shown that the problems are intractable, and thus formulation of approximation algorithms is useful. Next in Section 3.3, approximations are presented based on semidefinite relaxations of the quadratic integer formulations shown in Section 2.7. Section 3.3.1 explains how the the quadratic integer formulations are relaxed. Section 3.3.2 shows how the relaxed objectives can be represented in matrix form. Section 3.3.3 shows how the relaxed constraints can be represented in matrix form. Then Section 3.3.4 presents the semidefinite matrix formulations for the canonical set problems described in in Chapter 2. Section 3.3.5 explains the rounding process that is used to obtain approximate solutions. Section 3.3.6 presents proofs of performance bounds which provide guarantees on the quality of the solutions obtained using the SDP approximations. Finally, Section 3.4 presents quadratic approximations, an alternative method of approximation which remedies some of the drawbacks found in SDP approximations.

3.2 Proofs of Intractability

Problems are considered to be intractable if the computational effort of the best known algorithms to solve them grows exponentially as the size of the input is increased. To prove that an algorithm, $\mathcal{L}$, is intractable it suffices to show that there is a polynomial reduction from $\mathcal{L}'$, a problem that is known to be intractable, to $\mathcal{L}$. Such a reduction is known as a Karp reduction.

3.2.1 MDMC is Intractable

The minimum dominating maximum cut (MDMC) canonical subset problem described in Section 2.7.1 is reiterated for clarity:

(MDMC):

$$\begin{aligned}
\text{Maximize} \quad & \lambda_1 \left(\frac{1}{2} \sum_{i=1}^{n} (1 - y_i y_{n+1}) \right) && (3.1)\\
& +\lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} (1 - y_i y_j) \right), && (3.2)\\
\text{Subject to} \quad & (1 + y_i y_{n+1}) + \sum_{j=1}^{n} \mathcal{A}_{ij} (1 + y_j y_{n+1}) \geq 2, \ \forall\, 1 \leq i \leq n, \\
& y_i \in \{-1, +1\}, \ \forall\, 1 \leq i \leq n+1,
\end{aligned}$$

where λ_1 and λ_2 are non-negative and $\lambda_1 + \lambda_2 = 1$.

An instance of MDMC can be described as $\mathcal{L} = \{G(V, E), \Lambda\}$ where G is an undirected edge weighted graph, and $\Lambda = \{\lambda_1, \lambda_2\}$, is the set of weighting parameters for the objective functions. An optimal solution, $\mathcal{P}^*$, to this integer program is a vector $\mathbf{y} = [y_1, ..., y_{n+1}]^T$, indicating which vertices belong to the MDMC subset.

Theorem 3.2.1. *The minimum dominating maximum cut (MDMC) canonical subset problem is NP-hard.*

Proof. With respect to the set Λ, there are three cases to consider.

Case 1. $\lambda_1 = 0, \lambda_2 = 1$.

In this case the value of objective (3.1) is zero, and the problem is identical to MAX-CUT [30] with the additional constraint that the solution must be a dominating set. A proof by contradiction shows that an optimal solution $\mathcal{P}^*$ to the MDMC problem, $\mathcal{L} = \{G, \Lambda\}$, where $\Lambda = \{\lambda_1 = 0, \lambda_2 = 1\}$ is an optimal solution to MAX-CUT.

Assume that an optimal solution to MDMC is not an optimal solution to MAX-CUT. Then at least one vertex, v, exists in $\mathcal{P}^*$ or $\mathcal{P} \setminus \mathcal{P}^*$ which can be removed, which will result in a greater cut value, *i.e.* objective (3.2) would be larger. This implies that if $v \in \mathcal{P}^*$, there exists at least one intra edge, or if $v \in \mathcal{P} \setminus \mathcal{P}^*$ there exists at least one extra edge. But in either case switching v to the other subset will not violate the dominating set constraint because in the case of $v \in \mathcal{P}^*$, the vertex on the other side of the intra edge is in the canonical set, and in the case of $v \in \mathcal{P} \setminus \mathcal{P}^*$, vertex v will be put in the canonical set. This contradicts the original assumption that the solution was an optimal solution to MDMC.

Case 2. $\lambda_1 = 1, \lambda_2 = 0.$

In this case the value of objective (3.2) is zero, and the problem is identical to an instance of the minimum dominating set (rewritten as maximizing the inverse partition). It can be shown that the minimum dominating set problem is NP-Hard through a reduction from minimum set cover [3].

Case 3. $\lambda_1 > 0, \lambda_2 > 0.$

Observe that the contribution of a single vertex to the combined objective is λ_1, while the contribution of a single edge is $\lambda_2 \mathcal{W}_{ij}$, where $\mathcal{W}_{ij}$ is the weight of the edge. By setting the edge weights to sufficiently small values, the contribution of objective (3.2) will be rendered inconsequential. As a result an instance of the minimum dominating set, given by the graph G', may be reduced to an instance of MDMC. Let $\epsilon < \frac{\lambda_1}{\lambda_2 |E|}$, where $|E|$ denotes the number of edges in graph G'. Construct a new graph G from G' and set the weight of each edge to ϵ. Clearly this can be done in polynomial time. Then a proof by contradiction shows that a solution to the MDMC problem, $\mathcal{L} = \{G = (V, E), \Lambda\}$, is a minimum dominating set.

Assume that $\mathcal{P}^*$ is an optimal solution to the MDMC problem, and $\mathcal{L} = \{G(V, E), \Lambda\}$ is not a minimum dominating set. By definition it is a dominating set, so at least one vertex,

v, exists in $\mathcal{P}^*$ which can be removed. In addition, there must be at least one intra edge. If this were not the case, removal of vertex v would violate the dominating set constraint. The removal of vertex v from $\mathcal{P}^*$ will increase the value of objective (3.1) by λ_1 and decrease the value of objective (3.2) by at most $\epsilon\lambda_2(|E|-2)$. This implies

$$\begin{aligned}\epsilon\lambda_2(|E|-2) &= \frac{\lambda_2\lambda_1(|E|-2)}{\lambda_2|E|}\\ &= \frac{\lambda_1(|E|-2)}{|E|}\\ &< \lambda_1.\end{aligned}$$

That is, the removal of vertex v will always cause the combined objective to have a higher value, irrespective of the edges that are adjacent to v. But if this is the case, $\mathcal{P}^*$ is not optimal, which is a contradiction.

□

3.2.2 BCS is Intractable

The bounded canonical set (BCS) problem described in Section 2.7.2 is reiterated for clarity,

(BCS):

$$\text{Maximize} \quad \lambda_1 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - y_i y_j) + \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - y_i y_{n+1})(1 - y_j y_{n+1}) \right) \tag{3.3}$$

$$+ \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - y_i y_j) \right), \tag{3.4}$$

$$\text{Subject to} \quad \frac{1}{2} \sum_{i=1}^{n} (1 + y_i y_{n+1}) - k_{min} \geq 0,$$

$$k_{max} - \frac{1}{2} \sum_{i=1}^{n} (1 + y_i y_{n+1}) \geq 0,$$

$$y_i \in \{-1, +1\}, \ \forall\, 1 \leq i \leq n + 1,$$

where λ_1 and λ_2 are non-negative and $\lambda_1 + \lambda_2 = 1$.

An instance of BCS can be described as $\mathcal{L} = \{G = (V, E), \Lambda\}$ where G is an undirected edge weighted graph, and $\Lambda = \{\lambda_1, \lambda_2\}$, is the set of weighting parameters for the objective functions.

Lemma 3.2.2. *Let $G = (V, E)$ be a graph with edge weights of 1, and let c be the size of the minimum dominating set of G. If $\mathcal{L} = \{G = (V, E), \Lambda\}, \ \lambda_1 > 0, \lambda_2 > 0, \ k_{min} = 0, \ c \leq k_{max} \leq |V|$ is an instance of the BCS problem and all of the vertices of G have degree greater than zero, then $\mathcal{P}^*$ is a dominating set.*

Proof. Assume not, then there exists a vertex v in $\mathcal{P} \setminus \mathcal{P}^*$ which has no neighbors in $\mathcal{P}^*$. But if that were true then $\mathcal{P}^*$ would not be an optimal solution to BCS, since moving vertex v into $\mathcal{P}^*$ would increase the value of objective term (3.4). Objective term (3.3) will not change since any edges that were extra edges will become cut edges. □

Theorem 3.2.3. *The bounded canonical canonical subset problem (BCS) is NP-hard if $\lambda_1 > 0, \lambda_2 > 0$.*

Proof. An instance of the minimum dominating set may be reduced to an instance of BCS in the following manner. Given a graph $G' = (V', E')$, construct a new graph G from all of the vertices in G' with degree greater than zero along with all of the edges, E'. Clearly this can be accomplished in polynomial time.

With the fact that a solution to the BCS problem is a dominating set by Lemma 3.2.2, it can be shown that the minimum dominating set problem on G could be solved by adjusting the bound, k_{max}, and iterating through at most $|V|$ instances of BCS. Once a solution is found add the vertices $V' \setminus V$ to $\mathcal{P}^*$, thereby constructing the minimum dominating set of graph G'. □

3.2.3 SBCS is Intractable

Consider the stable bounded canonical set problem (see Section 2.7.3):

(SBCS):

$$\text{Maximize} \quad \lambda_1 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - y_i y_j) + \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - y_i y_{n+1})(1 - y_j y_{n+1}) \right) \tag{3.5}$$

$$+ \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - y_i y_j) \right) \tag{3.6}$$

$$+ \lambda_3 \left(\frac{1}{2} \sum_{i=1}^{n} t_i(1 + y_i y_{n+1}) \right), \tag{3.7}$$

$$\text{Subject to} \quad \frac{1}{2} \sum_{i=1}^{n} (1 + y_i y_{n+1}) - k_{min} \geq 0,$$

$$k_{max} - \frac{1}{2} \sum_{i=1}^{n} (1 + y_i y_{n+1}) \geq 0,$$

$$y_i \in \{-1, +1\}, \ \forall \, 1 \leq i \leq n+1.$$

The following proof shows that SBCS is intractable.

Theorem 3.2.4. *The stable bounded canonical canonical subset problem (SBCS) is NP-*

hard for all positive λ_1, λ_2, *and* λ_3.

Proof. An instance of the minimum dominating set may be reduced to an instance of SBCS in the following manner. Assign a weight of 1 to each edge of the graph, $\mathcal{W}_{ij} = 1,\ (u_i, v_j) \in E$. Then let $t_i = 1 \ \forall\ u_i \in V$. Clearly this can be accomplished in polynomial time. Next it is argued that a solution to the SBCS problem is a dominating set in the following way; Assume not, then there exists a vertex v in $\mathcal{P} \setminus \mathcal{P}^*$ which has no neighbors in $\mathcal{P}^*$. But if that were true then $\mathcal{P}^*$ would not be a SBCS since moving the vertex v into $\mathcal{P}^*$ would increase objective term (3.7) and possibly also objective term (3.6) if the vertex had edges. Objective term (3.5) will not change since any edges that were extra edges will become cut edges.

With the fact that a solution to the SBCS problem is a dominating set, it can be shown that the minimum dominating set problem could be solved by adjusting the bounds, k_{min}, k_{max}, and iterating through $|V|$ instances of SBCS.

□

3.3 Semidefinite Programming Approximation

In order to find an approximate solution to the problems described in Section 2.7, they can be relaxed and then formulated as semidefinite programs. Semidefinite programming is a generalization of linear programming [27] in the sense that linear programs can be formulated as semidefinite programs.

Consider the standard formulation of a linear program,

$$\begin{aligned} \text{Maximize} \quad & c^T x \\ \text{Subject to} \quad & a_i^T x = b_i, i = 1, \ldots, m \\ & x \in \Re^n_+, \end{aligned}$$

where c, a_i, b_i, x are vectors with x unknown and the objective is to maximize a linear vector inner-product form $c^T x = \sum_i^n c_i^T x_i$ over the polyhedral feasible region obtained by the intersection of constraints $a_i^T x = b_i$, $i = 1, \ldots, m$ and positive orthant $x \in \Re_+^n$. The standard form of a semidefinite program is

$$\begin{aligned} \textbf{(SDP):}\ \ \text{Maximize} \quad & \mathcal{C} \bullet \mathcal{X} \\ \text{Subject to} \quad & \mathcal{D}_i \bullet \mathcal{X} \geq 0,\ \forall\, i = 1 \ldots, m, \\ & \mathcal{X} \succeq 0, \end{aligned}$$

where $\mathcal{C}, \mathcal{D}_i, \mathcal{X}$ are $n \times n$ matrices with $\mathcal{X}$ unknown and where $\mathcal{C} \bullet \mathcal{X}$ denotes the Frobenius inner product of matrices $\mathcal{C}$ and $\mathcal{X}$, *i.e.*, $\mathcal{C} \bullet \mathcal{X} = \textbf{Trace}(\mathcal{C}^T \mathcal{X})$, and $\mathcal{X} \succeq 0$ means the matrix $\mathcal{X}$ is a positive semidefinite matrix.

In contrast to a linear programming problem, the objective function of a semidefinite program has a matrix inner-product form of $\mathcal{C} \bullet \mathcal{X}$. The set of feasible solutions for this optimization consists of the polyhedral resulting from the intersection of m linear constraints $\mathcal{D}_i \bullet \mathcal{X} \geq 0$ and the cone of positive semidefinite matrices $\{\mathcal{X} \succeq 0\}$. Simply put, the objective is encoded into the matrix $\mathcal{C}$, and the m constraints are encoded into the matrices $\mathcal{D}_1, \ldots, \mathcal{D}_m$.

Like linear programs, semidefinite programs also have duals. In fact most of the strong and weak duality results of linear programming have counterparts in semidefinite programming [1]. The dual of the semidefinite program 3.8 can be stated as:

$$\begin{aligned} \textbf{(SDP Dual):}\ \ \text{Maximize} \quad & \sum_{i=1}^{m} y_i b_i \\ \text{Subject to} \quad & \sum_{i=1}^{m} y_i \mathcal{D}_i + \mathcal{S} = \mathcal{C}, \\ & \mathcal{S} \succeq 0. \end{aligned}$$

Semidefinite programs can be solved in polynomial time assuming a polynomial representation of the input, and interior point methods have been shown to perform well in practice [29, 111].

3.3.1 Vector Relaxation of Canonical Set Problems

To find an approximate solution to the problems described in Section 2.7, the integral constraints are relaxed. Each indicator variable, y_i, $1 \leq i \leq n+1$, is replaced with a vector $x_i \in S_{n+1}$, where S_{n+1} is the unit sphere in $\mathbb{R}^{n+1}$. The relaxed form of the MDMC problem described in Section 2.7.1 is written as

(MDMC): (3.8)

$$\text{Maximize} \quad \lambda_1 \left(\frac{1}{2} \sum_{i=1}^{n} (1 - x_i^T x_{n+1}) \right) + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} (1 - x_i^T x_j) \right),$$

$$\text{Subject to} \quad (1 + x_i^T x_{n+1}) + \sum_{j=1}^{n} \mathcal{A}_{ij} (1 + x_j^T x_{n+1}) \geq 2, \ \forall\, 1 \leq i \leq n, \tag{3.9}$$

$$x_i^T x_i = 1, \ 1 \leq i \leq n+1, x_i \in \mathbb{R}^{n+1}, \tag{3.10}$$

where λ_1 and λ_2 are non-negative and $\lambda_1 + \lambda_2 = 1$. The relaxed form of the BCS problem described in Section 2.7.2 is written as

(BCS): (3.11)

$$\begin{aligned}
\text{Maximize} \quad & \lambda_1 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - x_i^T x_j) + \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - x_i^T x_{n+1})(1 - x_j^T x_{n+1}) \right) \\
& \quad + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - x_i^T x_j) \right), \\
\text{Subject to} \quad & \frac{1}{2} \sum_{i=1}^{n} (1 + x_i^T x_{n+1}) - k_{min} \geq 0, \qquad (3.12) \\
& k_{max} - \frac{1}{2} \sum_{i=1}^{n} (1 + x_i^T x_{n+1}) \geq 0, \\
& x_i^T x_i = 1,\ 1 \leq i \leq n+1, x_i \in \mathbb{R}^{n+1}, \qquad (3.13)
\end{aligned}$$

where λ_1 and λ_2 are non-negative and $\lambda_1 + \lambda_2 = 1$. The relaxed form of the SBCS problem described in Section 2.7.3 is written as

(SBCS): (3.14)

$$\begin{aligned}
\text{Maximize} \quad & \lambda_1 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - x_i^T x_j) + \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - x_i^T x_{n+1})(1 - x_j^T x_{n+1}) \right) \\
& \quad + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - x_i^T x_j) \right) + \lambda_3 \left(\frac{1}{2} \sum_{i=1}^{n} t_i (1 + x_i^T x_{n+1}) \right), \\
\text{Subject to} \quad & \frac{1}{2} \sum_{i=1}^{n} (1 + x_i^T x_{n+1}) - k_{min} \geq 0, \qquad (3.15) \\
& k_{max} - \frac{1}{2} \sum_{i=1}^{n} (1 + x_i^T x_{n+1}) \geq 0, \\
& x_i^T x_i = 1,\ 1 \leq i \leq n+1, x_i \in \mathbb{R}^{n+1}, \qquad (3.16)
\end{aligned}$$

where λ_1, λ_2, and λ_3 are non-negative and $\sum_{m=1}^{3} \lambda_m = 1$.

3.3.2 SDP Property Matrix Formulation

To express objective functions in the form of $\mathcal{C} \bullet \mathcal{X}$, define the matrix $\mathcal{V}$ to be the matrix obtained by concatenating the column vectors x_i, $1 \leq i \leq n+1$, and define $\mathcal{X} = \mathcal{V}^T\mathcal{V}$. Clearly, $\mathcal{X}$ is a semidefinite matrix, since $y^T\mathcal{X}y = y^T\mathcal{V}^T\mathcal{V}y = ||\mathcal{V}y||_2^2,\ \forall y \in \mathbb{R}^{n+1}$. Next, note some facts about the Frobenius inner product of matrices;

$$\begin{aligned} \mathcal{C}_1 \bullet \mathcal{X} + \mathcal{C}_2 \bullet \mathcal{X} &= \mathbf{Trace}(\mathcal{C}_1^T\mathcal{X}) + \mathbf{Trace}(\mathcal{C}_2^T\mathcal{X}) && (3.17)\\ &= \mathbf{Trace}((\mathcal{C}_1 + \mathcal{C}_2)^T\mathcal{X}) && (3.18)\\ &= (\mathcal{C}_1 + \mathcal{C}_2) \bullet \mathcal{X}. && (3.19) \end{aligned}$$

In order to facilitate the formulation of the canonical set algorithms, the properties described in Tables 2.1 and 2.2 are expressed in relaxed matrix form in Tables 3.1 and 3.2 respectively.

To see how this is done, consider the relaxation of the Size($\mathcal{P}^*$) property,

$$\frac{1}{2}\sum_{i=1}^{n}(1 + x_i^T x_{n+1}), \tag{3.20}$$

which can be written as $\mathcal{C} \bullet \mathcal{X}$ where

$$\mathcal{C} = \begin{bmatrix} \tilde{\mathbf{0}} & \frac{1}{4}\mathbf{e} \\ \frac{1}{4}\mathbf{e}^T & \frac{1}{2}n \end{bmatrix},$$

$\mathbf{e}$ is the all-ones vector of order n, and $\tilde{\mathbf{0}}$ is an $n \times n$ matrix of zeros.

The remaining properties can be transformed in a similar way.

Property	**Formulation**	**Description**
Size($\mathcal{P}^*$)	$\begin{bmatrix} \tilde{\mathbf{0}} & \frac{1}{4}\mathbf{e} \\ \frac{1}{4}\mathbf{e}^T & \frac{1}{2}n \end{bmatrix}$	Cardinality ($\lvert\mathcal{P}^*\rvert$) of canonical set
Stability($\mathcal{P}^*$)	$\begin{bmatrix} \tilde{\mathbf{0}} & \frac{1}{4}\mathbf{t} \\ \frac{1}{4}\mathbf{t}^T & \frac{1}{2}t_\Sigma \end{bmatrix}$	Stability of canonical set
Cut($\mathcal{P}^*$)	$\begin{bmatrix} -\frac{1}{4}\mathcal{W} & \hat{\mathbf{0}} \\ \hat{\mathbf{0}}^T & \frac{1}{4}w_\Sigma \end{bmatrix}$	Sum of cut edge weights
Intra($\mathcal{P}^*$)	$\begin{bmatrix} \frac{1}{4}\mathcal{W} & \frac{1}{4}\mathbf{d} \\ \frac{1}{4}\mathbf{d}^T & \frac{1}{4}w_\Sigma \end{bmatrix}$	Sum of intra edge weights
Extra($\mathcal{P}^*$)	$\begin{bmatrix} \frac{1}{4}\mathcal{W} & -\frac{1}{4}\mathbf{d} \\ -\frac{1}{4}\mathbf{d}^T & \frac{1}{4}w_\Sigma \end{bmatrix}$	Sum of extra edge weights

Table 3.1: Properties of canonical sets expressed in matrix form, where $\tilde{\mathbf{0}}$ is an $n \times n$ matrix of zeros, $\mathbf{e}$ is the all-ones vector of order n, $\mathbf{t}$ is a column vector in $\mathbb{R}^n$ whose i^{th} entry is t_i, $t_\Sigma = \sum_{i=1}^n t_i$, $\mathcal{W}$ is the $n \times n$ similarity matrix, $w_\Sigma = \sum_{i,j} \mathcal{W}_{ij}$, $\hat{\mathbf{0}}$ is an all zeros column vector in $\mathbb{R}^n$, and $\mathbf{d}$ is a column vector in $\mathbb{R}^n$ whose i^{th} entry has value $d_i = \sum_{j=1}^n \mathcal{W}_{ij}$.

Property	Formulation	Description
Size$^{-1}(\mathcal{P}^*)$	$\begin{bmatrix} \tilde{\mathbf{0}} & -\frac{1}{4}\mathbf{e} \\ -\frac{1}{4}\mathbf{e}^T & \frac{1}{2}n \end{bmatrix}$	$\lvert P \setminus P^* \rvert$
Stability$^{-1}(\mathcal{P}^*)$	$\begin{bmatrix} \tilde{\mathbf{0}} & -\frac{1}{4}\mathbf{t} \\ -\frac{1}{4}\mathbf{t}^T & \frac{1}{2}t_\Sigma \end{bmatrix}$	Stability of $P \setminus P^*$
Cut$^{-1}(\mathcal{P}^*)$	$\begin{bmatrix} \frac{1}{4}\mathcal{W} & \hat{\mathbf{0}} \\ \hat{\mathbf{0}}^T & \frac{1}{4}w_\Sigma \end{bmatrix}$	Sum of uncut edge weights
Intra$^{-1}(\mathcal{P}^*)$	$\begin{bmatrix} \tilde{\mathbf{0}} & -\frac{1}{4}\mathbf{d} \\ -\frac{1}{4}\mathbf{d}^T & \frac{1}{2}w_\Sigma \end{bmatrix}$	Sum of non-intra edge weights
Extra$^{-1}(\mathcal{P}^*)$	$\begin{bmatrix} \tilde{\mathbf{0}} & \frac{1}{4}\mathbf{d} \\ \frac{1}{4}\mathbf{d}^T & \frac{1}{2}w_\Sigma \end{bmatrix}$	Sum of non-extra edge weights

Table 3.2: Property inverses of canonical sets expressed in matrix form, where $\tilde{\mathbf{0}}$ is an $n \times n$ matrix of zeros, $\mathbf{e}$ is the all-ones vector of order n, $\mathbf{t}$ is a column vector in $\mathbb{R}^n$ whose i^{th} entry is t_i, $t_\Sigma = \sum_{i=1}^{n} t_i$, $\mathcal{W}$ is the $n \times n$ similarity matrix, $w_\Sigma = \sum_{i,j} \mathcal{W}_{ij}$, $\hat{\mathbf{0}}$ is an all zeros column vector in $\mathbb{R}^n$, and $\mathbf{d}$ is a column vector in $\mathbb{R}^n$ whose i^{th} entry has value $d_i = \sum_{j=1}^{n} \mathcal{W}_{ij}$.

3.3.3 SDP Constraint Matrix Formulation

The constraints described in Section 2.5 can also be formulated as matrices. For example, the dominating set constraint (3.9),

$$(1 + x_i^T x_{n+1}) + \sum_{j=1}^{n} \mathcal{A}_{ij}(1 + x_j^T x_{n+1}) \geq 2, \ \forall\, 1 \leq i \leq n,$$

may be written as

$$x_i^T x_{n+1} + \sum_{j=1}^{n} \mathcal{A}_{ij}(x_j^T x_{n+1}) \ \ \geq \ \ 1 - d_i,$$

where d_i is the degree of vertex i. This may be written as n constraint matrices, $\mathcal{D}_i$, for $i = 1, \ldots, n$, where each matrix has the form

$$\mathcal{D}_i = \begin{bmatrix} 0 & 0 & \ldots & 0 & 0 & \ldots & \mathcal{A}_{i,1} \\ 0 & 0 & \ldots & 0 & 0 & \ldots & \mathcal{A}_{i,2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 1 \\ 0 & 0 & \ldots & 0 & 0 & \ldots & \mathcal{A}_{i,i+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & \mathcal{A}_{i,n} \\ \mathcal{A}_{i,1} & \mathcal{A}_{i,2} & \ldots & 1 & \mathcal{A}_{i,i+1} & \ldots & 2(d_i - 1) \end{bmatrix}. \tag{3.21}$$

Constraints (3.10), (3.13), and (3.16),

$$x_i^T x_i = 1, \ 1 \leq i \leq n+1, x_i \in \mathbb{R}^{n+1},$$

may be encoded as $n+1$ constraint matrices, $\mathcal{D}_1, \ldots, \mathcal{D}_{n+1}$, that are all zeros with a single 1 that moves along the main diagonal, enforcing the $x_i^T x_i = 1$ constraints. For example,

$$\mathcal{D}_1 = \begin{bmatrix} 1 & \hat{\mathbf{0}} \\ \hat{\mathbf{0}}^T & \tilde{\mathbf{0}} \end{bmatrix}, \mathcal{D}_{n+1} = \begin{bmatrix} \tilde{\mathbf{0}} & \hat{\mathbf{0}} \\ \hat{\mathbf{0}}^T & 1 \end{bmatrix}, \tag{3.22}$$

where $\tilde{\mathbf{0}}$ is an $n \times n$ matrix of zeros, and $\hat{\mathbf{0}}$ is an all zeros column vector in $\mathbb{R}^n$. Constraints (3.12) and (3.15),

$$\frac{1}{2}\sum_{i=1}^{n}(1 + x_i^T x_{n+1}) - k_{min} \geq 0,$$

may be written as

$$\mathcal{D} = \begin{bmatrix} \tilde{\mathbf{0}} & \frac{1}{4}\mathbf{e} \\ \frac{1}{4}\mathbf{e}^T & \frac{1}{2}n - k_{min} \end{bmatrix}, \tag{3.23}$$

where $\mathbf{e}$ is the all-ones vector of order n. In a similar way, constraints (3.12) and (3.15),

$$k_{max} - \frac{1}{2}\sum_{i=1}^{n}(1 + x_i^T x_{n+1}) \geq 0,$$

may be expressed as

$$\mathcal{D} = \begin{bmatrix} \tilde{\mathbf{0}} & \frac{1}{4}\mathbf{e} \\ \frac{1}{4}\mathbf{e}^T & k_{max} - \frac{1}{2}n \end{bmatrix}. \tag{3.24}$$

3.3.4 SDP Matrix Formulation

Recall that the MDMC problem described in Section 2.7.1 is formulated as a maximization of Size$^{-1}(\mathcal{P}^*)$ and Cut$(\mathcal{P}^*)$. Matrix formulations from Tables 3.1 and 3.2 may be used to write the SDP form of the MDMC problem described in Section 3.8 as

$$\begin{aligned} &\textbf{(MDMC):} && \qquad (3.25)\\ &\text{Maximize} && \mathcal{C} \bullet \mathcal{X} \\ &\text{Subject to} && \mathcal{D}_i \bullet \mathcal{X} \geq 0,\ \forall\, i = 1 \dots, m, \\ & && \mathcal{X} \succeq 0, \end{aligned}$$

where

$$\mathcal{C} \;=\; \lambda_1 \begin{bmatrix} \tilde{\mathbf{0}} & -\frac{1}{4}\mathbf{e} \\ -\frac{1}{4}\mathbf{e}^T & \frac{1}{2}n \end{bmatrix} + \lambda_2 \begin{bmatrix} -\frac{1}{4}\mathcal{W} & \hat{\mathbf{0}} \\ \hat{\mathbf{0}}^T & \frac{1}{4}w_\Sigma \end{bmatrix},$$

$\mathcal{D}_0, \dots, \mathcal{D}_n$ are as described in (3.21), $\mathcal{D}_{n+1}, \dots, \mathcal{D}_{2n+1}$ are as described in (3.22), and $m = 2n + 1$. The coefficients, $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$ are parameters such that $\lambda_1 + \lambda_2 = 1$.

Recall also that the BCS problem described in Section 2.7.2 is formulated as a maximization of Intra$^{-1}(\mathcal{P}^*)$ and Cut$(\mathcal{P}^*)$. Using formulations from Tables 3.1 and 3.2, the SDP form of the BCS problem described in Section 3.11 is written as

$$\begin{aligned} &\textbf{(BCS):} && \qquad (3.26)\\ &\text{Maximize} && \mathcal{C} \bullet \mathcal{X} \\ &\text{Subject to} && \mathcal{D}_i \bullet \mathcal{X} \geq 0,\ \forall\, i = 1 \dots, m, \\ & && \mathcal{X} \succeq 0, \end{aligned}$$

where

$$\mathcal{C} \;=\; \lambda_1 \begin{bmatrix} \tilde{\mathbf{0}} & -\frac{1}{4}\mathbf{d} \\ -\frac{1}{4}\mathbf{d}^T & \frac{1}{2}w_\Sigma \end{bmatrix} + \lambda_2 \begin{bmatrix} -\frac{1}{4}\mathcal{W} & \hat{\mathbf{0}} \\ \hat{\mathbf{0}}^T & \frac{1}{4}w_\Sigma \end{bmatrix},$$

$\mathcal{D}_1, \ldots, \mathcal{D}_{n+1}$ are as described in (3.22), $\mathcal{D}_{n+2}$ is as described in (3.23), $\mathcal{D}_{n+3}$ is as described in (3.24), and $m = n + 3$. The coefficients, $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$ are parameters such that $\lambda_1 + \lambda_2 = 1$.

Note also that the SBCS problem described in Section 2.7.3 is formulated as a maximization of Intra$^{-1}(\mathcal{P}^*)$, Cut($\mathcal{P}^*$), and Stability($\mathcal{P}^*$). Again using formulations from Tables 3.1 and 3.2, the SDP form of the SBCS problem described in Section 3.14 is written as

$$\begin{aligned} &\textbf{(SBCS):} && \qquad (3.27) \\ &\text{Maximize} && \mathcal{C} \bullet \mathcal{X} \\ &\text{Subject to} && \mathcal{D}_i \bullet \mathcal{X} \geq 0,\ \forall\, i = 1 \ldots, m, \\ & && \mathcal{X} \succeq 0, \end{aligned}$$

where

$$\mathcal{C} = \lambda_1 \begin{bmatrix} \tilde{\mathbf{0}} & -\frac{1}{4}\mathbf{d} \\ -\frac{1}{4}\mathbf{d}^T & \frac{1}{2}w_\Sigma \end{bmatrix} + \lambda_2 \begin{bmatrix} -\frac{1}{4}\mathcal{W} & \hat{\mathbf{0}} \\ \hat{\mathbf{0}}^T & \frac{1}{4}w_\Sigma \end{bmatrix} + \lambda_3 \begin{bmatrix} \tilde{\mathbf{0}} & \frac{1}{4}\mathbf{t} \\ \frac{1}{4}\mathbf{t}^T & \frac{1}{2}t_\Sigma \end{bmatrix},$$

$\mathcal{D}_1, \ldots, \mathcal{D}_{n+1}$ are as described in (3.22), $\mathcal{D}_{n+2}$ is as described in (3.23), $\mathcal{D}_{n+3}$ is as described in (3.24), and $m = n + 3$. The coefficients, $\lambda_1 \geq 0, \lambda_2 \geq 0$, and $\lambda_3 \geq 0$ satisfy $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

3.3.5 Rounding

Once the solutions to the semidefinite programs described in Section 3.3.4 are computed, a rounding step must be performed to obtain an integer solution. The rounding method detailed by Vazirani [112] is used to identify the set of values for indicator variables $y_1, ..., y_n$, and the set indicator variable y_{n+1}. This method is based on Cholesky decomposition and a multivariate normal hyperplane method that can be effectively derandomized [79].

Let $\mathcal{X}^* = \mathcal{X}^*(\Lambda)$ denote the optimal solution of the canonical set obtained from the semidefinite program. Since $\mathcal{X}^*$ is a symmetric positive semidefinite matrix, using Cholesky decomposition it can be represented as $\mathcal{X}^* = \mathcal{V}^T\mathcal{V}$ [32]. This provides us with $n+1$ vectors for the relaxed canonical set problems. Specifically, column x_i, $1 \leq i \leq n$, of $\mathcal{V}$ forms the vector associated with vertex $v_i \in G$ in the optimal SDP relaxation of the canonical set problem, and column x_{n+1} corresponds to the set indicator variable. Finally

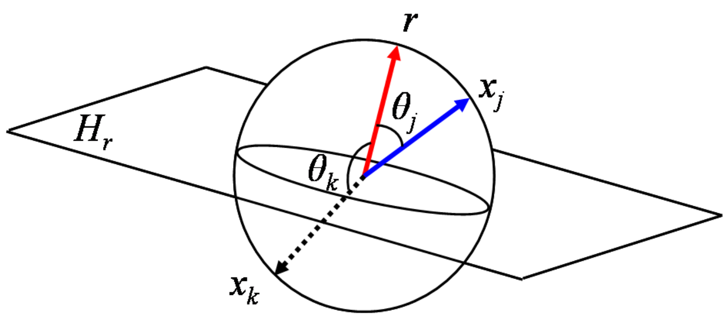


Figure 3.1: Random vector r on the unit sphere.

pick a random vector, $r \in \mathbb{R}^{n+1}$, uniformly distributed on the unit sphere. For each column x_i, $i \leq 1 \leq n+1$ of $\mathcal{V}$ generate the indicator variable y_i such that

$$y_i = \begin{cases} +1 & \text{if } r^T x_i \geq 0 \\ -1 & \text{otherwise.} \end{cases}$$

Denote the canonical set $\mathcal{P}^* = \{v_i \,|\, y_i = y_{n+1}, 1 \leq i \leq n\}$.

This rounding step is performed multiple times, each time checking that the constraints are not violated, and keeping track of the solution with the best objective value.

3.3.6 Bounds

During the rounding step described in Section 3.3.5, a random vector ρ is used to determine set assignment. The value of a rounded solution can be stated in terms of the probability of the separation of vectors $x_i, x_j \in \mathbb{R}^{n+1}$ by the random vector ρ.

Lemma 3.3.1 (Vazirani (lemma 26.6) [112])**.**

$$\mathbf{Pr}[x_i \textit{ and } x_j \textit{ are separated}] = \frac{\theta_{i,j}}{\pi},$$

where $\theta_{i,j}$ is the angle between x_i and x_j. This is the probability of data points p_i and p_j being in different sets. Let $\theta' = \pi - \theta$, noting that

$$\mathbf{Pr}[x_i \textit{ and } x_j \textit{ are not separated}] = \frac{\theta'_{i,j}}{\pi}.$$

A lemma from Goemans and Williamson [31] is used.

Lemma 3.3.2 (Goemans and Williamson (lemma 2.5) [31])**.** *Let*

$$\alpha = \min_{0<\theta'\leq\pi} \frac{2}{\pi}\frac{\theta'}{(1-\cos\theta')},$$

then $\alpha \geq .87856$.

This can be seen by letting

$$f(\theta') = \frac{2}{\pi}\frac{\theta'}{(1-\cos\theta')} \tag{3.28}$$

and then computing its derivative,

$$f'(\theta') = \frac{2}{\pi}\left(\frac{1}{1-\cos\theta'} - \frac{\theta'\sin\theta'}{(1-\cos\theta')^2}\right).$$

Then for $f'(\theta') = 0$, $\theta' \approx -2.33$, which is then plugged into equation (3.28). Note

Lemma 3.3.3. *For all* θ' *such that* $0 \leq \theta' \leq \pi$

$$\frac{\theta'}{\pi} \geq \alpha \left(\frac{1 - \cos\theta'}{2} \right).$$

This can be seen given the definition of α and noting

$$\frac{\theta'}{\pi} = \frac{2}{\pi} \cdot \frac{\theta'}{1 - \cos\theta'} \cdot \frac{1 - \cos\theta'}{2}.$$

MDMC Bounds

Next it is shown that the algorithm for approximating the MDMC is at least 0.878 of optimal for a fixed set $\Lambda = \{\lambda_1, \lambda_2\}$. To do this, the expected value of the approximate solution is computed and shown to be at least 0.878 of optimal.

The maximization MDMC objective,

$$\lambda_1 \left(\frac{1}{2} \sum_{i=1}^{n} (1 - x_i^T x_{n+1}) \right) + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} (1 - x_i^T x_j) \right),$$

can be restated in terms of θ, noting the vectors x_i, x_j, and x_{n+1} are unit vectors. Their contribution to the objective is

$$\lambda_1 \left(\frac{1}{2} \sum_{i=1}^{n} (1 - \cos\theta_{i,n+1}) \right) + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} (1 - \cos\theta_{i,j}) \right).$$

The objective value of the optimal solution to MDMC may be defined as

$$OPT_{MDMC} = \lambda_1 \left(\frac{1}{2} \sum_{i=1}^{n} (1 - \cos\theta_{i,n+1}) \right) + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} (1 - \cos\theta_{i,j}) \right). \quad (3.29)$$

Next the expected value of the approximate solution is found and shown that it is at least

$0.878 \cdot OPT_{MDMC}$ using Lemmas 3.3.2 and 3.3.3.

Let Z_{MDMC} denote the value of the objective (3.29) obtained using the rounding method described in Section 3.3.5. Note that $0 \leq 1 - \cos\theta \leq 2$, so the expected value is $2 \cdot \frac{\theta}{\pi}$ using Lemma 3.3.1. Also note that $\frac{\theta}{\pi} \geq \alpha\left(\frac{1-\cos\theta}{2}\right)$ from Lemma 3.3.3 using a change of variables. Then the expectation of Z_{MDMC} is

$$\begin{aligned}
\mathbf{E}[Z_{MDMC}] &= \lambda_1\left(\frac{1}{2}\sum_{i=1}^{n}\left(2 \cdot \frac{\theta_{i,n+1}}{\pi}\right)\right) + \lambda_2\left(\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}\left(2 \cdot \frac{\theta_{i,j}}{\pi}\right)\right) \\
&\geq \alpha \cdot \left[\lambda_1\left(\frac{1}{2}\sum_{i=1}^{n}\left(2 \cdot \frac{(1-\cos\theta_{i,n+1})}{2}\right)\right)\right. \\
&\quad \left. +\lambda_2\left(\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}\left(2 \cdot \frac{(1-\cos\theta_{i,j})}{2}\right)\right)\right] \\
&= \alpha \cdot \left[\lambda_1\left(\frac{1}{2}\sum_{i=1}^{n}(1-\cos\theta_{i,n+1})\right)\right. \\
&\quad \left. +\lambda_2\left(\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-\cos\theta_{i,j})\right)\right] \\
&= \alpha \cdot OPT_{MDMC}.
\end{aligned}$$

Thus it has been shown that for any fixed $\Lambda = \{\lambda_1, \lambda_2\}$, the expected value of the approximate solution is at least $0.878 \cdot OPT_{MDMC}$, where OPT_{MDMC} is the optimal solution to the MDMC problem.

BCS Bounds

A similar argument holds for approximating the BCS which shows the expected value of the approximate solution is at least 0.878 of optimal.

The maximization BCS objective,

$$\text{Maximize} \qquad \lambda_1 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - x_i^T x_j) + \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - x_i^T x_{n+1})(1 - x_j^T x_{n+1}) \right) + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - x_i^T x_j) \right), \tag{3.30}$$

can be restated in terms of θ. Their contribution to the objective is

$$\lambda_1 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - \cos\theta_{i,j}) + \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - \cos\theta_{i,n+1})(1 - \cos\theta_{j,n+1}) \right) + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - \cos\theta_{i,j}) \right). \tag{3.31}$$

Next the expected value of the approximate solution is found and shown that it is at least $0.878 \cdot OPT_{BCS}$ using Lemmas 3.3.2 and 3.3.3.

Let Z_{BCS} denote the value of the objective (3.31) obtained using the rounding method described in Section 3.3.5. Again note that $0 \leq 1 - \cos\theta \leq 2$, so the expected value is $2 \cdot \frac{\theta}{\pi}$

using Lemma 3.3.1. Then the expectation of Z_{BCS} is

$$\begin{aligned}
\mathbf{E}[Z_{BCS}] &= \lambda_1 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} \left(2 \cdot \frac{\theta_{i,j}}{\pi} \right) + \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} \left(2 \cdot \frac{\theta_{i,n+1}}{\pi} \right) \left(2 \cdot \frac{\theta_{j,n+1}}{\pi} \right) \right) \\
&\quad + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} \left(2 \cdot \frac{\theta_{i,j}}{\pi} \right) \right) \\
&\geq \alpha \cdot \Bigg[\lambda_1 \Bigg(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} \left(2 \cdot \frac{(1 - \cos \theta_{i,j})}{2} \right) \\
&\quad + \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} \left(2 \cdot \frac{(1 - \cos \theta_{i,n+1})}{2} \right) \left(2 \cdot \frac{(1 - \cos \theta_{j,n+1})}{2} \right) \Bigg) \\
&\quad + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} \left(2 \cdot \frac{(1 - \cos \theta_{i,j})}{2} \right) \right) \Bigg] \\
&= \alpha \cdot \Bigg[\lambda_1 \Bigg(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} \left(1 - \cos \theta_{i,j}\right) \\
&\quad + \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} \left(1 - \cos \theta_{i,n+1}\right) \left(1 - \cos \theta_{j,n+1}\right) \Bigg) \\
&\quad + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} \left(1 - \cos \theta_{i,j}\right) \right) \Bigg] \\
&= \alpha \cdot OPT_{BCS}.
\end{aligned}$$

Thus it has been shown that for any particular any particular set $\Lambda = \{\lambda_1, \lambda_2\}$, the expected value of the approximate solution is at least $0.878 \cdot OPT_{BCS}$, where OPT_{BCS} is the optimal solution to the BCS problem. It is important to note that the rounding process may violate the size constraints k_{min} and k_{max}. The guarantee only applies to the objective value.

SBCS Bounds

Using a similar argument, it can be shown that the algorithm for approximating the SBCS is at least 0.878 of optimal for any particular set $\Lambda = \{\lambda_1, \lambda_2, \lambda_3\}$.

The maximization SBCS objective,

$$\begin{aligned} \text{Maximize} \quad & \lambda_1 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - x_i^T x_j) + \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - x_i^T x_{n+1})(1 - x_j^T x_{n+1}) \right) \\ & + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - x_i^T x_j) \right) + \lambda_3 \left(\frac{1}{2} \sum_{i=1}^{n} t_i (1 + x_i^T x_{n+1}) \right), \end{aligned} \tag{3.32}$$

can then be restated in terms of θ. Their contribution to the objective is

$$\begin{aligned} & \lambda_1 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - \cos\theta_{i,j}) + \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - \cos\theta_{i,n+1})(1 - \cos\theta_{j,n+1}) \right) \\ & + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - \cos\theta_{i,j}) \right) + \lambda_3 \left(\frac{1}{2} \sum_{i=1}^{n} t_i (1 + \cos\theta_{i,n+1}) \right) \end{aligned} \tag{3.33}$$

Using the facts that $\theta' = \pi - \theta$ and $1 + \cos\theta = 1 - \cos\theta'$, the objective value of optimal solution to SBCS may be defined as

$$\begin{aligned} OPT_{SBCS} = \; & \lambda_1 \Bigg(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - \cos\theta_{i,j}) \\ & + \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - \cos\theta_{i,n+1})(1 - \cos\theta_{j,n+1}) \Bigg) \\ & + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - \cos\theta_{i,j}) \right) \\ & + \lambda_3 \left(\frac{1}{2} \sum_{i=1}^{n} t_i (1 - \cos\theta'_{i,n+1}) \right) \end{aligned} \tag{3.34}$$

Next the expected value of the approximate solution is found and shown that it is at least $0.878 \cdot OPT_{SBCS}$ using Lemmas 3.3.2 and 3.3.3.

Let Z_{SBCS} denote the value of the objective (3.34) obtained using the rounding method described in Section 3.3.5. Note that $0 \leq 1 - \cos\theta' \leq 2$ and $0 \leq 1 - \cos\theta \leq 2$, so the expected values are $2 \cdot \frac{\theta'}{\pi}$ and $2 \cdot \frac{\theta}{\pi}$ respectively using Lemma 3.3.1. Then the expectation of Z_{SBCS} is

$$\begin{aligned}
\mathbf{E}[Z_{SBCS}] &= \lambda_1\Bigg(\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}\left(2\cdot\frac{\theta_{i,j}}{\pi}\right)+\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}\left(2\cdot\frac{\theta_{i,n+1}}{\pi}\right)\left(2\cdot\frac{\theta_{j,n+1}}{\pi}\right)\Bigg) \\
&\quad +\lambda_2\left(\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}\left(2\cdot\frac{\theta_{i,j}}{\pi}\right)\right)+\lambda_3\left(\frac{1}{2}\sum_{i=1}^{n}t_i\left(2\cdot\frac{\theta'_{i,n+1}}{\pi}\right)\right) \qquad (3.35) \\
&\geq \alpha\cdot\Bigg[\lambda_1\Bigg(\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}\left(2\cdot\frac{(1-\cos\theta_{i,j})}{2}\right) \\
&\quad +\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}\left(2\cdot\frac{(1-\cos\theta_{i,n+1})}{2}\right)\left(2\cdot\frac{(1-\cos\theta_{j,n+1})}{2}\right)\Bigg) \\
&\quad +\lambda_2\left(\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}\left(2\cdot\frac{(1-\cos\theta_{i,j})}{2}\right)\right) \\
&\quad +\lambda_3\left(\frac{1}{2}\sum_{i=1}^{n}t_i\left(2\cdot\frac{(1-\cos\theta'_{i,n+1})}{2}\right)\right)\Bigg] \qquad (3.36) \\
&= \alpha\cdot\Bigg[\lambda_1\Bigg(\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}\left(1-\cos\theta_{i,j}\right) \\
&\quad +\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}\left(1-\cos\theta_{i,n+1}\right)\left(1-\cos\theta_{j,n+1}\right)\Bigg) \\
&\quad +\lambda_2\left(\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}\left(1-\cos\theta_{i,j}\right)\right) \\
&\quad +\lambda_3\left(\frac{1}{2}\sum_{i=1}^{n}t_i\left(1-\cos\theta'_{i,n+1}\right)\right)\Bigg] \qquad (3.37) \\
&= \alpha\cdot OPT_{SBCS}. \qquad (3.38)
\end{aligned}$$

Thus it has been shown that for any particular any particular set $\Lambda = \{\lambda_1, \lambda_2, \lambda_3\}$, the expected value of the approximate solution is at least $0.878 \cdot OPT_{SBCS}$, where OPT_{SBCS} is the optimal solution to the SBCS problem. Again, it is important to note the rounding process may violate the size constraints k_{min} and k_{max}, and the guarantee only applies to the objective value.

3.4 Quadratic Programming (QP) Approximation

One drawback to the SDP formulations for the canonical set described in Section 3.3 is that as the size of the input set increases, the number of variables in the SDP formulation increases quadratically. This is a direct result of the relaxation process where each indicator variable is replaced with a vector. One way to avoid this problem is to use quadratic programming approximations. Quadratic programs often perform well in practice, but their theoretic bounds can be difficult to determine. Nevertheless, this drawback is acceptable in situations where memory or runtime constraints exist.

In order find an approximate solution to the problems described in Section 2.7, the problems are formulated as quadratic programs. Consider a standard formulation of a quadratic program,

QP:

$$\begin{aligned} \text{Maximize} \quad & \frac{1}{2}x^T H x + f^T x && (3.39) \\ \text{Subject to} \quad & Ax \leq b, && (3.40) \\ & l \leq x \leq u, && (3.41) \end{aligned}$$

where the objective is quadratic and the constraints are linear. In this formulation, the objectives are encoded into the matrix H which is positive definite and the vector f. The constraints are encoded into the matrix A and the vectors b, l, u.

3.4.1 QP Property Formulation

To formulate the canonical set problems in quadratic form, the indicator variables are redefined as

$$y_i = \begin{cases} 1 & \text{if } p_i \in \mathcal{P}^* \\ -1 & \text{otherwise,} \end{cases}$$

and the set indicator is removed. In order to form an approximation, the integrality constraints on the indicator variables are removed and the value is allow to lie in the interval $[-1, 1]$. The property formulations of Table 2.1 are reformulated as shown in Table 3.3. In a similar manner Table 2.2 can be stated as Table 3.4.

Property	**Formulation**	**Description**
Size($\mathcal{P}^*$)	$\frac{1}{2}\sum_{i=1}^{n}(1+y_i)$	Cardinality ($\lvert\mathcal{P}^*\rvert$) of canonical set
Stability($\mathcal{P}^*$)	$\frac{1}{2}\sum_{i=1}^{n}t_i(1+y_i)$	Stability of canonical set
Cut($\mathcal{P}^*$)	$\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-y_iy_j)$	Sum of cut edge weights
Intra($\mathcal{P}^*$)	$\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1+y_i)(1+y_j)$	Sum of intra edge weights
Extra($\mathcal{P}^*$)	$\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-y_i)(1-y_j)$	Sum of extra edge weights

Table 3.3: Properties of canonical sets (QP)

Using the fact that $\mathcal{W}$ is symmetric where applicable, like terms can be combined and the properties rewritten as shown in Tables 3.5 and 3.6. Using these tables, the property and inverse property formulations for the quadratic approximation can be constructed by dropping the constants and formulating the quadratic terms in matrix form, and the linear terms in vector form as shown in Table 3.7.

Property	**Formulation**	**Description**
Size$^{-1}(\mathcal{P}^*)$	$\frac{1}{2}\sum_{i=1}^{n}(1-y_i)$	$\lvert P \setminus P^*\rvert$
Stability$^{-1}(\mathcal{P}^*)$	$\frac{1}{2}\sum_{i=1}^{n} t_i(1-y_i)$	Stability of $P \setminus P^*$
Cut$^{-1}(\mathcal{P}^*)$	$\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1+y_iy_j)$	Sum of uncut edge weights
Intra$^{-1}(\mathcal{P}^*)$	$\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-y_iy_j)$ $+\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-y_i)(1-y_j)$	Sum of non-intra edge weights
Extra$^{-1}(\mathcal{P}^*)$	$\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1-y_iy_j)$ $+\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}(1+y_i)(1+y_j)$	Sum of non-extra edge weights

Table 3.4: Property inverse formulations (QP)

Property	**Formulation**
Size$(\mathcal{P}^*)$	$\frac{n}{2}+\frac{1}{2}\sum_{i=1}^{n} y_i$
Stability$(\mathcal{P}^*)$	$\frac{1}{2}\sum_{i=1}^{n} t_i+\frac{1}{2}\sum_{i=1}^{n} t_iy_i$
Cut$(\mathcal{P}^*)$	$\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}-\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}y_iy_j$
Intra$(\mathcal{P}^*)$	$\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}+\frac{1}{2}\sum_{i=1}^{n} y_i\sum_{j=1}^{n}\mathcal{W}_{ij}+\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}y_iy_j$
Extra$(\mathcal{P}^*)$	$\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}-\frac{1}{2}\sum_{i=1}^{n} y_i\sum_{j=1}^{n}\mathcal{W}_{ij}+\frac{1}{4}\sum_{i,j}\mathcal{W}_{ij}y_iy_j$

Table 3.5: Properties of canonical sets (QP) combined

Property	**Formulation**
$\text{Size}^{-1}(\mathcal{P}^*)$	$\frac{n}{2} - \frac{1}{2}\sum_{i=1}^{n} y_i$
$\text{Stability}^{-1}(\mathcal{P}^*)$	$\frac{1}{2}\sum_{i=1}^{n} t_i - \frac{1}{2}\sum_{i=1}^{n} t_i y_i$
$\text{Cut}^{-1}(\mathcal{P}^*)$	$\frac{1}{4}\sum_{i,j} \mathcal{W}_{ij} + \frac{1}{4}\sum_{i,j} \mathcal{W}_{ij} y_i y_j$
$\text{Intra}^{-1}(\mathcal{P}^*)$	$\frac{1}{2}\sum_{i,j} \mathcal{W}_{ij} - \frac{1}{2}\sum_{i=1}^{n} y_i \sum_{j=1}^{n} \mathcal{W}_{ij}$
$\text{Extra}^{-1}(\mathcal{P}^*)$	$\frac{1}{2}\sum_{i,j} \mathcal{W}_{ij} + \frac{1}{2}\sum_{i=1}^{n} y_i \sum_{j=1}^{n} \mathcal{W}_{ij}$

Table 3.6: Property inverse formulations (QP) combined

Property	H	f	**Description**
$\text{Size}(\mathcal{P}^*)$	$[\tilde{\mathbf{0}}]$	$\frac{1}{2}\mathbf{e}$	Cardinality ($\lvert\mathcal{P}^*\rvert$) of canonical set
$\text{Stability}(\mathcal{P}^*)$	$[\tilde{\mathbf{0}}]$	$\frac{1}{2}\mathbf{t}$	Stability of canonical set
$\text{Cut}(\mathcal{P}^*)$	$[-\frac{1}{2}\mathcal{W}]$	$\hat{\mathbf{0}}$	Sum of cut edge weights
$\text{Intra}(\mathcal{P}^*)$	$[\frac{1}{2}\mathcal{W}]$	$\frac{1}{2}\mathbf{d}$	Sum of intra edge weights
$\text{Extra}(\mathcal{P}^*)$	$[\frac{1}{2}\mathcal{W}]$	$-\frac{1}{2}\mathbf{d}$	Sum of extra edge weights

Property	H	f	**Description**
$\text{Size}^{-1}(\mathcal{P}^*)$	$[\tilde{\mathbf{0}}]$	$-\frac{1}{2}\mathbf{e}$	$\lvert P \setminus P^*\rvert$
$\text{Stability}^{-1}(\mathcal{P}^*)$	$[\tilde{\mathbf{0}}]$	$-\frac{1}{2}\mathbf{t}$	Stability of $P \setminus P^*$
$\text{Cut}^{-1}(\mathcal{P}^*)$	$[\frac{1}{2}\mathcal{W}]$	$\hat{\mathbf{0}}$	Sum of uncut edge weights
$\text{Intra}^{-1}(\mathcal{P}^*)$	$[\tilde{\mathbf{0}}]$	$-\frac{1}{2}\mathbf{d}$	Sum of non-intra edge weights
$\text{Extra}^{-1}(\mathcal{P}^*)$	$[\tilde{\mathbf{0}}]$	$\frac{1}{2}\mathbf{d}$	Sum of non-extra edge weights

Table 3.7: QP Properties with inverses of canonical sets expressed in matrix and vector form, where $\tilde{\mathbf{0}}$ is an $n \times n$ matrix of zeros, $\mathbf{e}$ is the all-ones vector of order n, $\mathbf{t}$ is a column vector in $\mathbb{R}^n$ whose i^{th} entry is t_i, $\mathcal{W}$ is the $n \times n$ similarity matrix, $\hat{\mathbf{0}}$ is an all zeros column vector in $\mathbb{R}^n$, and $\mathbf{d}$ is a column vector in $\mathbb{R}^n$ whose i^{th} entry has value $d_i = \sum_{j=1}^{n} \mathcal{W}_{ij}$.

3.4.2 MDMC QP Formulation

Recall that the MDMC problem described in Section 2.7.1 is formulated as a maximization of Size$^{-1}(\mathcal{P}^*)$ and Cut($\mathcal{P}^*$). Matrix and vector formulations from Table 3.7 may be used to write the QP form of the MDMC problem described in Section 3.8 as

(MDMC QP):

$$\begin{array}{rl} \text{Maximize} & \frac{1}{2}x^T H x + f^T x \\ \text{Subject to} & Ax \leq b, \\ & l \leq x \leq u, \end{array}$$

where

$$\begin{aligned} H &= \frac{\lambda_2}{2}\left[\mathcal{W}\right], \\ A &= -(\mathcal{A} + I) \\ b &= \mathbf{d} - \mathbf{e} \\ f &= -\frac{\lambda_1}{2}\mathbf{e}, \\ l &= -\mathbf{e}, \\ u &= \mathbf{e}, \end{aligned}$$

$\mathcal{W}$ is the similarity matrix, $\mathcal{A}$ is the adjacency matrix, I is the identity matrix, $\mathbf{d}$ is a column vector in $\mathbb{R}^n$ whose i^{th} entry has value $d_i = \sum_{j=1}^{n} \mathcal{W}_{ij}$, and $\mathbf{e}$ is the all-ones vector of order n.

3.4.3 BCS QP Formulation

Recall also that the BCS problem described in Section 2.7.2 is formulated as a maximization of Intra$^{-1}(\mathcal{P}^*)$ and Cut($\mathcal{P}^*$). Using the matrix and vector formulations from

Table 3.7 the QP form of the BCS problem described in Section 3.8 may be written as

(BCS QP):

$$\begin{aligned} \text{Maximize} \quad & \frac{1}{2}x^T H x + f^T x \\ \text{Subject to} \quad & Ax \leq b, \\ & l \leq x \leq u, \end{aligned}$$

where

$$\begin{aligned} H &= -\frac{\lambda_2}{2}\left[\mathcal{W}\right], \\ A &= \begin{bmatrix} -\mathbf{e}^T \\ \mathbf{e}^T \end{bmatrix}, \\ b &= \begin{bmatrix} n - 2k_{min} \\ 2k_{max} - n \end{bmatrix}, \\ f &= -\frac{\lambda_1}{2}\mathbf{d}, \\ l &= -\mathbf{e}, \\ u &= \mathbf{e}, \end{aligned}$$

and $\mathcal{W}$ is the similarity matrix, $\mathbf{e}$ is the all-ones vector of order n, $\mathbf{d}$ is a column vector in $\mathbb{R}^n$ whose i^{th} entry has value $d_i = \sum_{j=1}^{n} \mathcal{W}_{ij}$, and $\mathbf{t}$ is a column vector in $\mathbb{R}^n$ whose i^{th} entry is t_i.

3.4.4 SBCS QP Formulation

The SBCS problem described in Section 2.7.3 is formulated as a maximization of Intra$^{-1}(\mathcal{P}^*)$, Cut($\mathcal{P}^*$), and Stability($\mathcal{P}^*$). Again using the matrix and vector formulations from Table 3.7 the QP form of the SBCS problem described in Section 3.8 may be written as

(SBCS QP):

$$\begin{aligned} \text{Maximize} \quad & \frac{1}{2}x^T H x + f^T x \\ \text{Subject to} \quad & Ax \leq b, \\ & l \leq x \leq u, \end{aligned}$$

where

$$\begin{aligned} H &= -\frac{\lambda_2}{2}\left[\mathcal{W}\right], \\ A &= \begin{bmatrix} -\mathbf{e}^T \\ \mathbf{e}^T \end{bmatrix}, \\ b &= \begin{bmatrix} n - 2k_{min} \\ 2k_{max} - n \end{bmatrix}, \\ f &= -\frac{\lambda_1}{2}\mathbf{d} + \frac{\lambda_3}{2}\mathbf{t}, \\ l &= -\mathbf{e}, \\ u &= \mathbf{e}, \end{aligned}$$

and $\mathcal{W}$ is the similarity matrix, $\mathbf{e}$ is the all-ones vector of order n, $\mathbf{d}$ is a column vector in $\mathbb{R}^n$ whose i^{th} entry has value $d_i = \sum_{j=1}^{n} \mathcal{W}_{ij}$, and $\mathbf{t}$ is a column vector in $\mathbb{R}^n$ whose i^{th} entry is t_i.

3.5 Conclusion

This chapter has presented approximation algorithms for the canonical set problems posed in Chapter 2. To motivate the need for approximations, it was shown in Section 3.2

that the canonical set problems are all NP-hard. Section 3.3 described semidefinite programming approximations for the canonical set problems. To create these approximation algorithms, it was first shown how the integer formulations from Chapter 2 can be relaxed. Then SDP formulations were presented and a rounding process was described which is used to obtain the approximate solutions. Proof of bounds were then given which established that the canonical set algorithms can achieve approximations that are at least 0.878 of optimal. Finally, in Section 3.4, quadratic programming approximation algorithms for the canonical sets described in Chapter 2 were given.

4. Simple and General Experimental Results

4.1 Introduction

This chapter presents experiments that first look at the question of how sensitive canonical sets are to outliers. Through a small set of experiments on synthetic datasets of collinear points, this question is examined. The sensitivity of the K-Means clustering algorithm to outliers was part of the motivation for the development of the canonical set method, so a comparison with K-Means is next presented. Finally, examples of canonical sets of some regular structures are presented to provide some intuition on the canonical set method.

4.2 Sensitivity to Outliers

To investigate the sensitivity of the canonical sets obtained using the formulations presented in Chapter 2, a small graph was first constructed which consisted of 5 vertices evenly spaced on a line. A complete graph was made using the inverse Euclidean distance for similarity. An exhaustive search version of the BCS algorithm was then run on the graph using parameters $\lambda_1 = 0.5,\ \lambda_2 = 0.5,\ k_{min} = 1$, and $k_{max} = 5$. The results of this experiment are shown in the top of Figure 4.1, with the canonical set being indicated with filled vertices. Then the rightmost vertex was progressively moved to the right, and the experiment was repeated as can be seen in the figure. Inspection of Figure 4.1 shows the canonical set remains the same as the vertex is moved. Next the BCS SDP approximation algorithm presented in Section 3.3 was run on the same dataset. The results were the same, confirming that the approximation is a good one, since in these cases the results are optimal.

To further investigate the case of 5 collinear points a complete graph was constructed (see Figure 4.2) where the distance between points was formulated algebraically. For technical reasons the canonical set algorithms are implemented as minimizations. Recall that

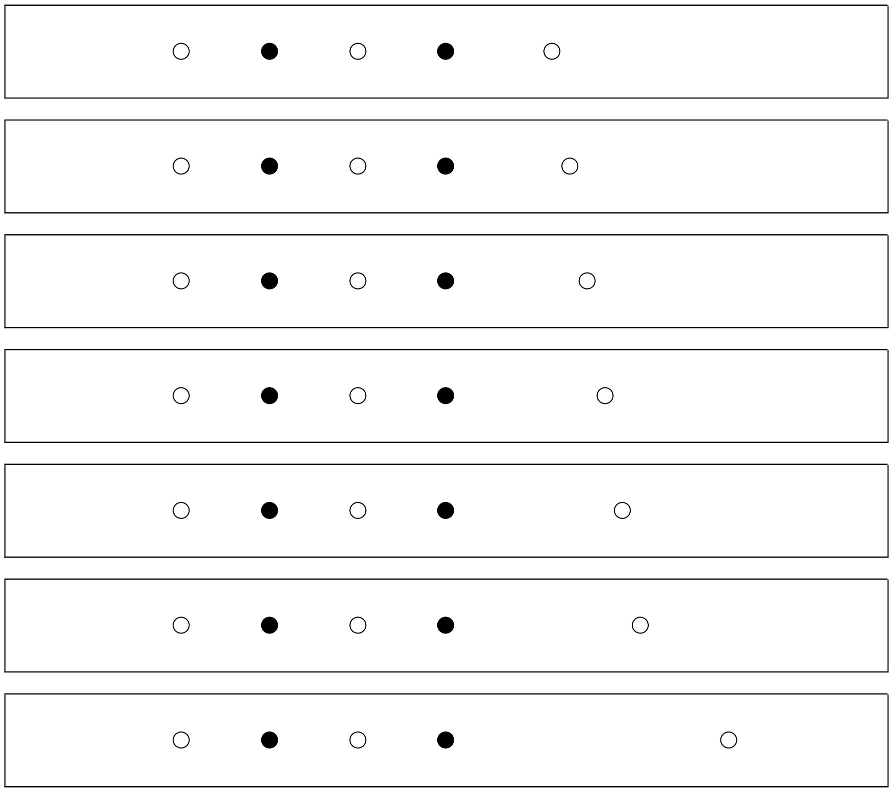


Figure 4.1: BCS of five collinear points as the rightmost point is progressively moved further to the right. The canonical set is marked as filled vertices.

the maximization objective of the BCS is

$$\lambda_1 \left[\text{Intra}^{-1}(\mathcal{P}^*)\right] + \lambda_2 \left[\text{Cut}(\mathcal{P}^*)\right].$$

The minimization form of the objective is

$$\begin{aligned} & \lambda_1 \left[\text{Intra}(\mathcal{P}^*)\right] + \lambda_2 \left[\text{Cut}^{-1}(\mathcal{P}^*)\right] \\ = \quad & \lambda_1 \left[\text{Intra}(\mathcal{P}^*)\right] + \lambda_2 \left[\text{Intra}(\mathcal{P}^*) + \text{Extra}(\mathcal{P}^*)\right]. \end{aligned}$$

The inverse Euclidean distance between the data points is used as the similarity measure.

Note that the canonical set of the graph given in Figure 4.2, consists of vertices 2 and 4.

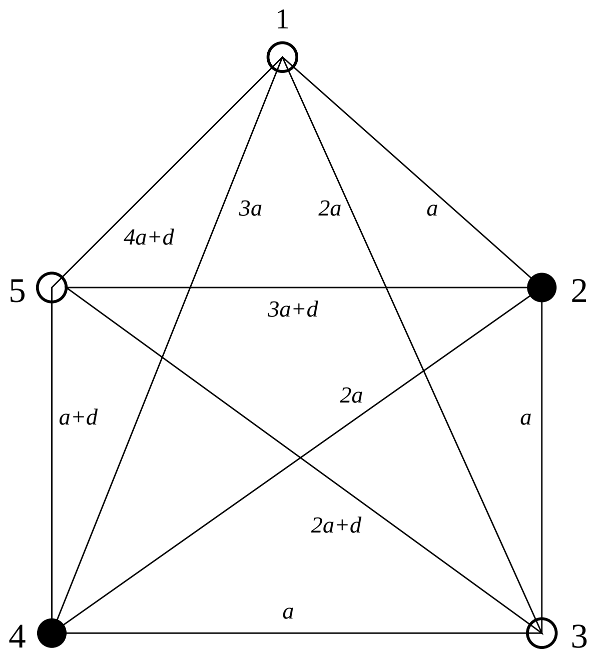


Figure 4.2: Complete graph of 5 collinear points used in the outlier sensitivity experiment with Euclidean distances. The canonical vertices are filled.

The objective value $\mathcal{M}(\{2,4\})$ may be written as

$$\begin{aligned}
\mathcal{M}(\{2,4\}) &= \lambda_1 \left[\text{Intra}(\{2,4\})\right] + \lambda_2 \left[\text{Intra}(\{2,4\}) + \text{Extra}(\{2,4\})\right] \\
&= \text{Intra}(\{2,4\}) + \frac{1}{2}\text{Extra}(\{2,4\}) \\
&= \frac{1}{2a} + \frac{\frac{1}{4a+d} + \frac{1}{2a} + \frac{1}{2a+d}}{2}.
\end{aligned}$$

A similar representation can be made for other possible subsets. Figure 4.3 shows a graph of the objective values of the canonical set (vertices 2 and 4) and all possible pairs of vertices which include the outlier (vertex 5) as the value of d (distance of vertex 5 from initial position) is increased. Inspection of the graph reveals why the canonical sets shown in Figure 4.1 do not change as vertex 5 is moved to the right. The objective value of the canonical set (vertices 2 and 4) is lower than any pair which contains the outlier (vertex 5).

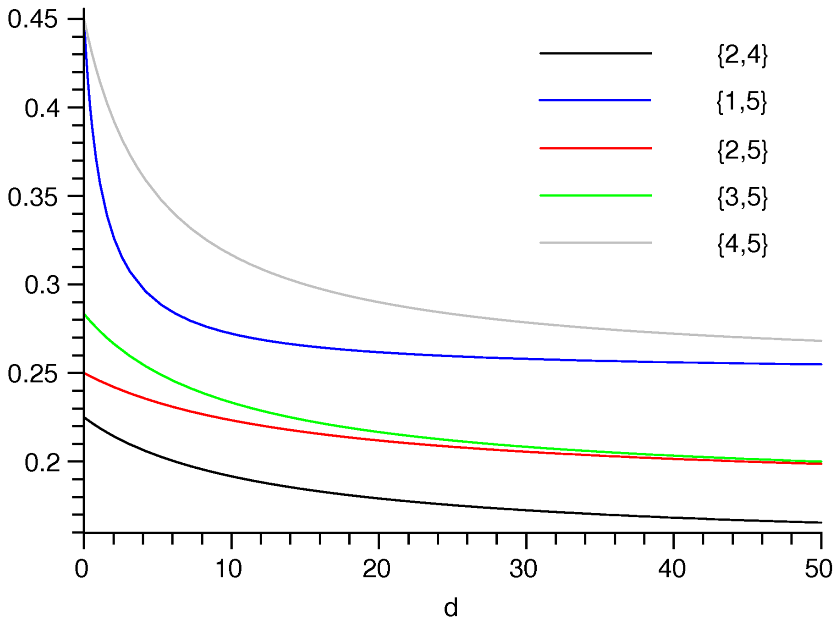


Figure 4.3: Graph of the objective values for subsets as the value of d is increased. As d is increased, vertex 5 moves further to the right. The graph shows that pairs which include the outlier (vertex 5) have higher objective values than the vertex pair of $\{2, 4\}$ which is the canonical set.

Figures 4.4 and 4.5 show similar experiments with graphs with 6 and 8 points respectively with $\lambda_1 = 0.5,\ \lambda_2 = 0.5,\ k_{min} = 1$, and $k_{max} = |V|$. Of particular note is the canonical set at the top of Figure 4.5, which has a canonical set of size 3. If $k_{max} = 2$, then the canonical set is the same as the others in Figure 4.5.

This set of experiments has shown that in the examples tested, the canonical set method is resistant to outliers. Moving a boundary point further away from the rest of the dataset does not affect the selected subset. In the case of 5 collinear points, a convincing case was presented which showed that no matter how far the boundary point was moved, the canonical set will not change (if the size is 2).

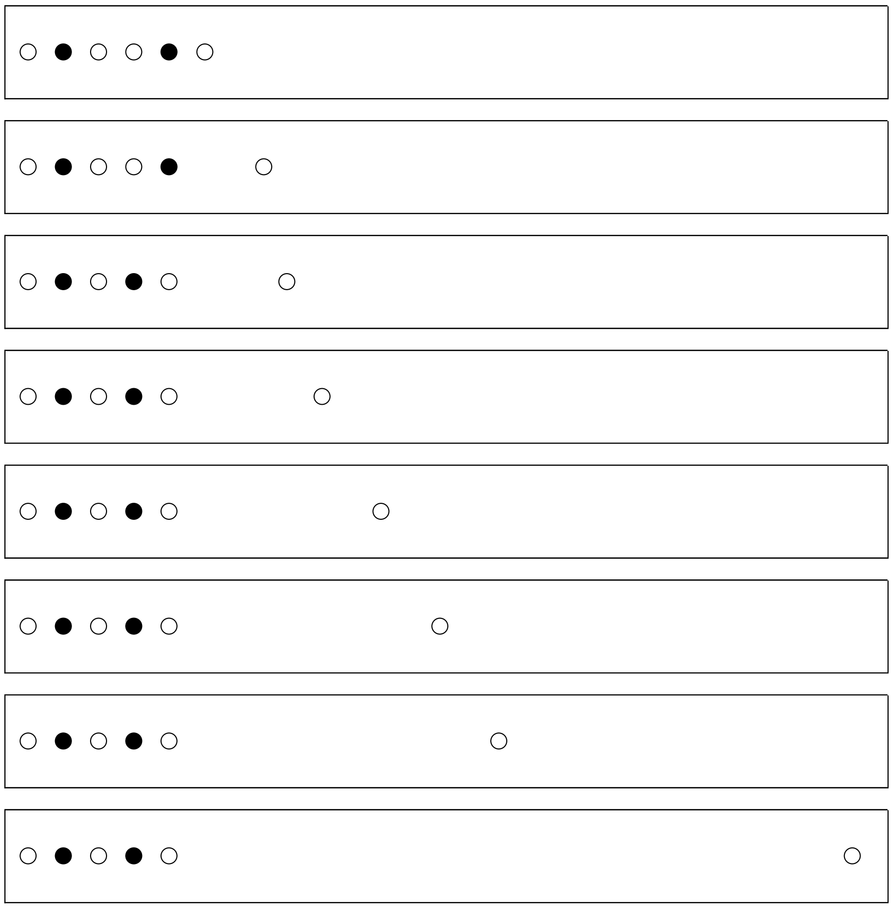


Figure 4.4: BCS of six collinear points.

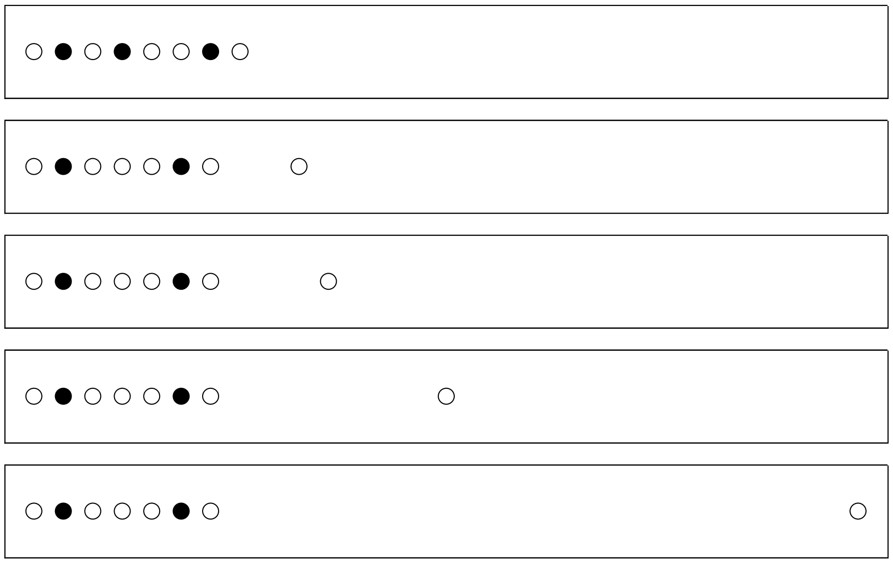


Figure 4.5: BCS of eight collinear points.

4.3 Comparison with K-Means

In this experiment sets of 5 collinear data points are again used. As before in Section 4.2, completely connected graphs are constructed with the edge weight being given by the inverse Euclidean distance between the vertices. The graphs are identical except that the rightmost vertex has been moved to the right in each successive graph.

Figure 4.6 shows the results of the BCS SDP approximation algorithm. The BCS SDP algorithm was run with $\lambda_1 = 0.5,\ \lambda_2 = 0.5,\ k_{min} = 1$, and $k_{max} = 5$. The canonical vertices are filled in black and the induced clustering is shown. Figure 4.7 shows the result of the K-Means algorithm with $k = 2$. In this case, if a centroid was located at a vertex of the graph it is filled in in black, otherwise the centroid location is indicated in red. The clustering (dashed groups) changes as the outlier moves to the right. If the data point nearest the cluster centroid is used, the results although different (see Figure 4.8) are also sensitive to the position of the outlier.

Why K-Means is so sensitive outliers, can be seen by examining Figure 4.7. In the top cluster on the right there are 2 data points. The centroid is calculated from these 2 points, so moving the rightmost point further right by some distance d will move the centroid to the right by $\frac{d}{2}$.

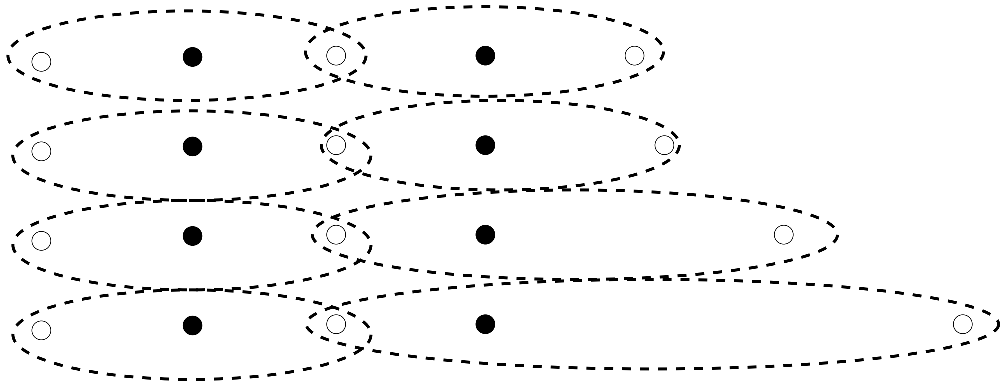


Figure 4.6: Four sets of 5 horizontally spaced data points. The BCS vertices are solid black. The induced clustering (dashed groups) is not affected by the outlier.

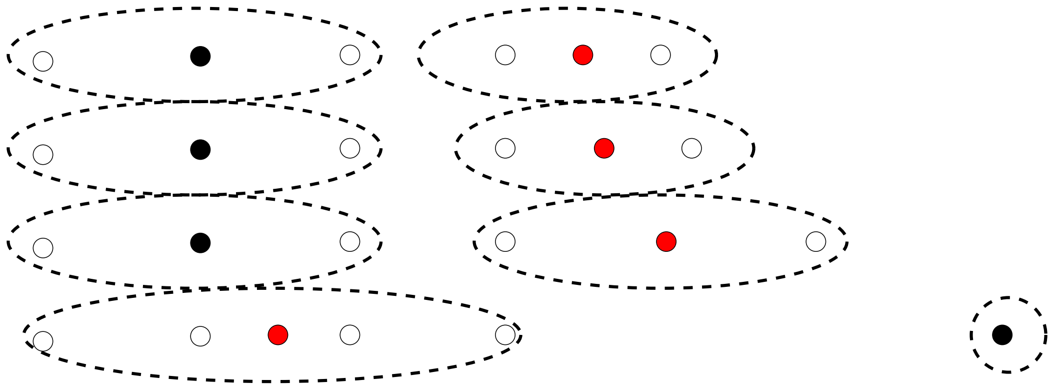


Figure 4.7: Four sets of 5 horizontally spaced data points. The K-Means centroids are solid black if they are at a data point location and red otherwise. The clustering (dashed groups) changes as the outlier moves to the right.

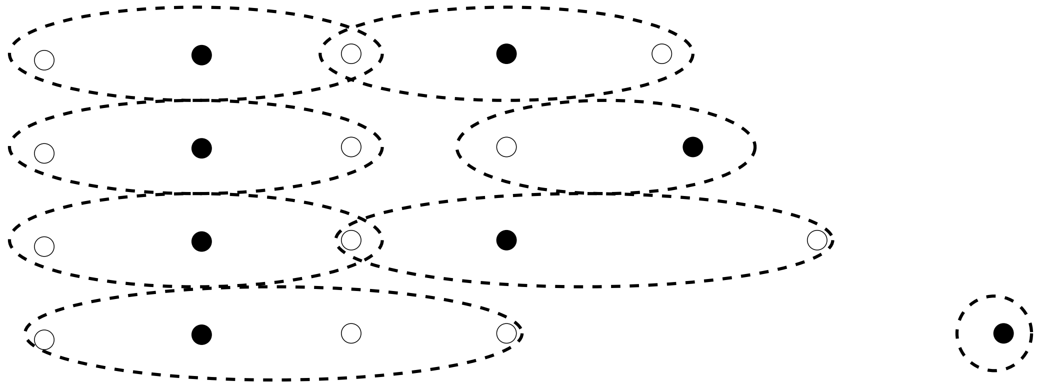


Figure 4.8: Four sets of 5 horizontally spaced data points. The K-Means centroids nearest neighbors are solid black. The clustering (dashed groups) changes as the outlier moves to the right.

4.4 Regular Structures

In this section, canonical subsets of some synthetic datasets are presented. Section 4.4.1 shows results from the BCS SDP approximation algorithm on trees, bipartite, spiral, grid, lattice, and clusters. Section 4.4.2 shows results from the MDMC SDP algorithm on a lattice and two trees.

4.4.1 BCS

The following subsets were created by the BCS SDP approximation algorithm with $\lambda_1 = 0.5$, $\lambda_2 = 0.5$, $k_{min} = 1$, and $k_{max} = |V|$ unless otherwise noted. The inverse Euclidean distance between the data points was used as a similarity measure.

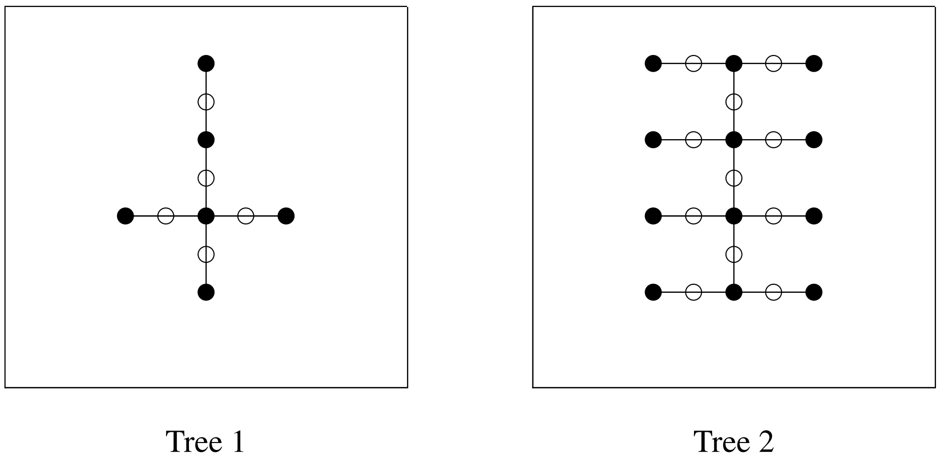
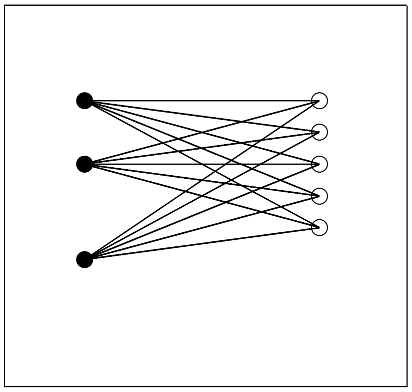
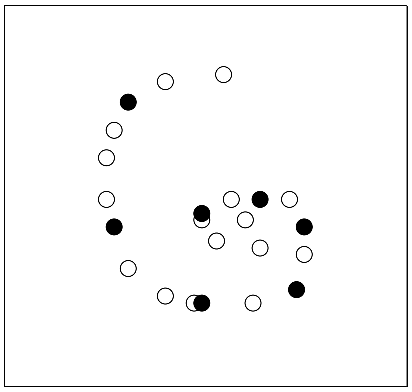


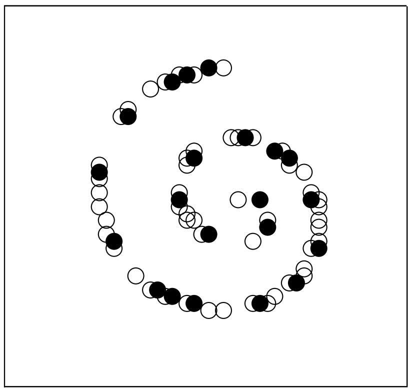
Figure 4.9: Sample trees and BCS



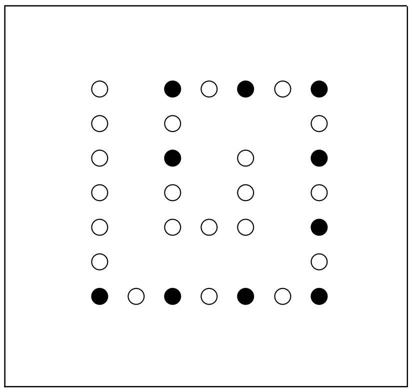
Bipartite graph $k_{max} = 3$



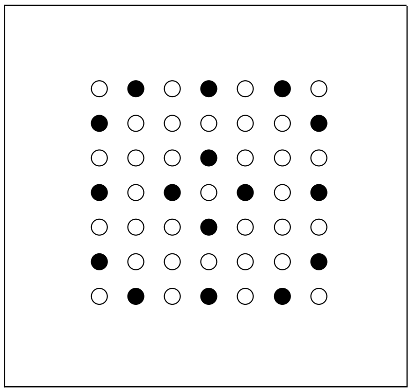
Spiral 1



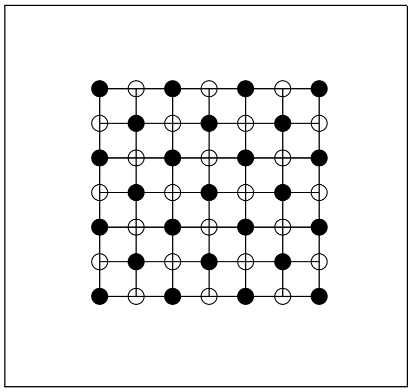
Spiral 2



Spiral 3

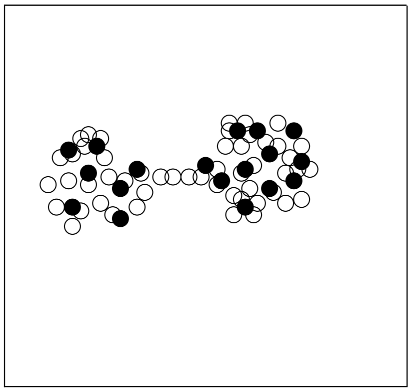


Fully connected grid

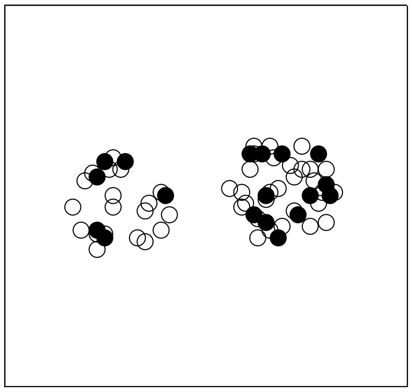


Four way connected lattice

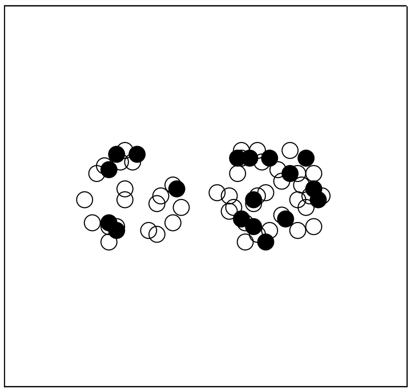
Figure 4.10: Sample graphs and BCS



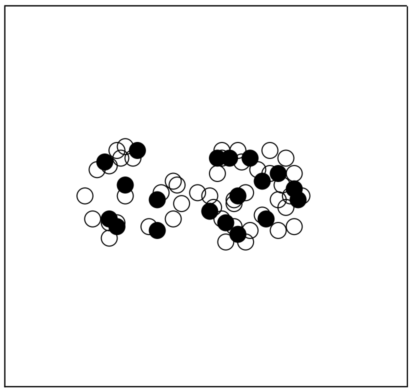
Clusters 1



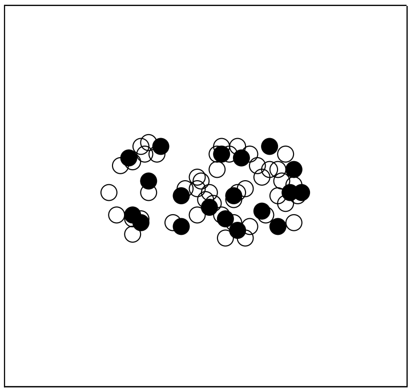
Clusters 2a



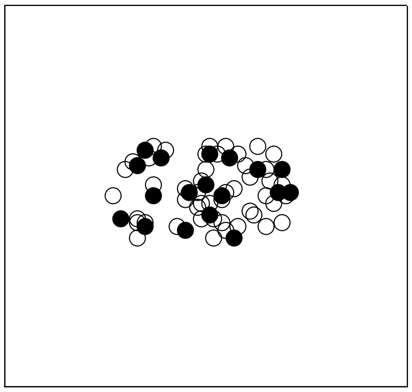
Clusters 2b



Clusters 2c



Clusters 2d

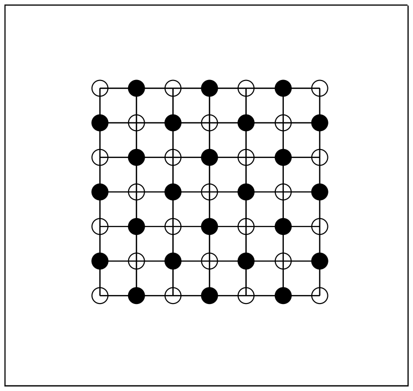


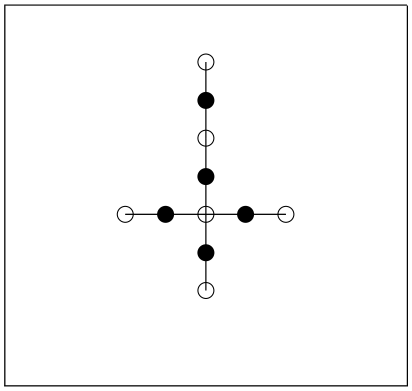
Clusters 2e

Figure 4.11: Cluster 1 with BCS marked by filled vertices. Clusters 2a-e show two clusters merging and the BCS at each step.

4.4.2 MDMC

The following subsets were created by the MDMC algorithm with $\lambda_1 = 0.5$ and $\lambda_2 = 0.5$. The inverse Euclidean distance between the data points was used as a similarity measure.


Four way connected lattice


Tree 1

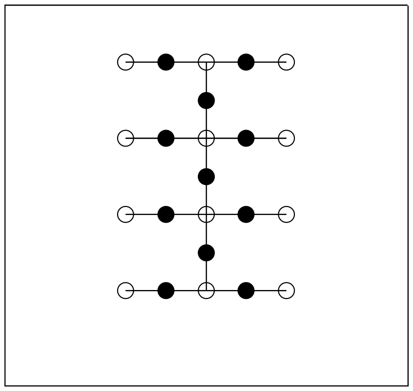

Tree 2

Figure 4.12: Sample graphs and MDMC

4.5 Conclusion

This chapter has presented preliminary experiments that evaluate the subsets selected by the algorithms presented in Chapters 2 and 3. These limited experiments showed canonical sets have low sensitivity to outliers. First a set of experiments on collinear data points showed this to be true. Next a comparison with K-Means was made showing that it was more sensitive to outliers than the canonical set method. Finally, the canonical sets of some example graphs was given to provide intuition on what canonical sets look like.

5. View Selection

5.1 Introduction

One problem in computer vision is that of 2D view selection. The objective is to identify highly informative 2D views of a 3D object. To motivate the problem, consider the 3D object in Figure 5.1(a). Assume a dense set of 2D views $\mathcal{P}$ of this object over the viewing sphere exists, such as in Figure 5.1(b), which shows 68 silhouettes obtained from this 3D object. The goal is to identify a small set of views that closely resembles the full set of views in $\mathcal{P}$.

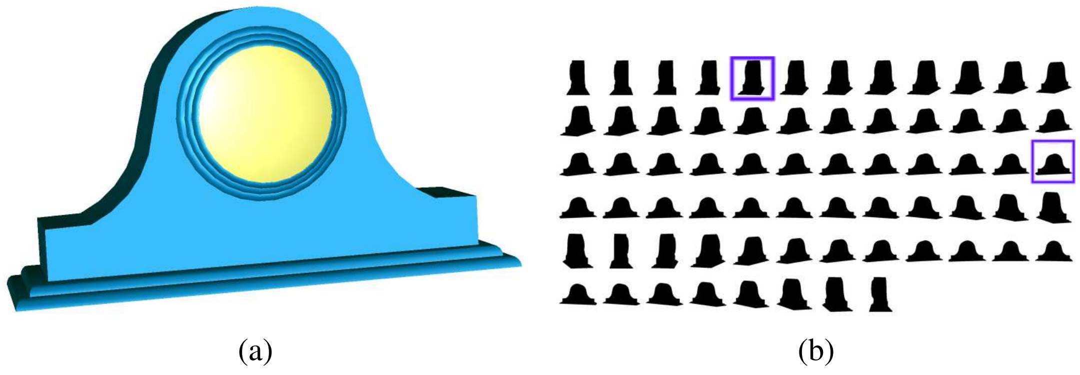


Figure 5.1: Motivation: Given the 3D model in (a) and a set of views in (b), and similarity function among the views, identify a small subset of views that best characterizes the object.

The idea is to select a canonical subset of views using algorithms described in Chapter 3. The motivation for this approach came from the novel view expressed by Cyr and Kimia [13] that "…the shape similarity metric between object outlines endows the viewing sphere with a metric which can be used to cluster views into aspects, and to represent each aspect with a prototypical view." These ideas were introduced in the context of aspect graph representations [54] and their relevance in identifying regions of “equivalent views” on the viewing sphere.

Intuitively, in the graph representation, each vertex represents an “aspect” of the 3D object, *i.e.*, a maximally connected region on the viewing sphere. The edges of this graph will correspond to visual transitions between two neighboring general views. Bowyer and Dyer [6] presented a recent survey on aspect graphs and their applications. To reduce the complexity of generating the aspects, Eggert *et al.* [21] developed the notion of a scale-space aspect graph, and Dickinson *et al.* [19] used a hierarchical aspect graph system based on a finite set of primitives. Weinshall and Werman [114] studied the notions of view stability and view likelihood to establish a theory which defines the aspect graph.

More recently, the identification of canonical views of 3D objects has been studied [76, 89]. These canonical views can then be used in technical drawings and computer visualizations. Canonical views are similar to the prototype views described by Cyr and Kimia [13], which were used for 3D object recognition.

In order to select a subset of views, a similarity measure is required so that there is a notion of distance between pairs of views. Section 5.2 describes the distance measure. Section 5.3 describes the view selection technique and the experiments using the minimum dominating max cut (MDMC) canonical subset algorithm (3.25), which was described in Chapter 3. For details the reader is referred to [17]. Finally, in Section 5.4 experiments using the bounded canonical set (BCS) algorithm for view-based 3D object recognition are presented. These results were published in [18].

5.2 Distance Measure

The algorithm used for computing the many-to-many matching and the distance between 2D views represented as silhouettes is that of Demirci *et al.* [15]. For a given view, an object’s silhouette is first represented by an undirected, rooted, weighted graph, in which nodes represent *shocks* [104] (or, equivalently, skeleton branches) and edges connect adjacent shock branches.

An illustration of this representation is given in Figure 5.2. The left portion shows the initial silhouette and its shock points (skeleton). The right portion depicts the constructed shock tree. Darker, heavier nodes correspond to fragments whose average radii are larger.

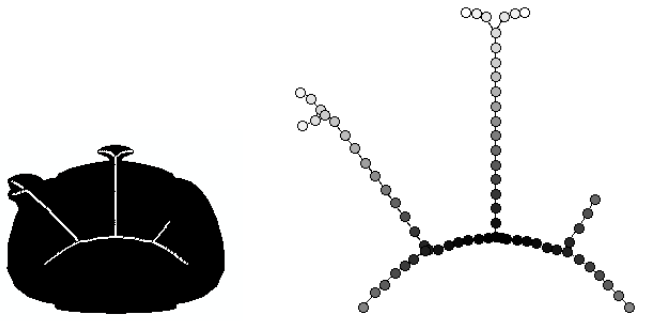


Figure 5.2: Left: the silhouette and its medial axis. Right: the medial axis tree constructed from the medial axis. Darker nodes reflect larger radii.

The matching algorithm is based on the metric-tree representation of labeled graphs and their low-distortion embeddings into normed vector spaces via spherical coding. This two-step transformation reduces the many-to-many matching problem to that of computing a distribution based distance measure between two such embeddings. To compute the distance between two sets of weighted vectors, the Earth Mover's Distance under transformation is used. For two given 2D views, the algorithm provides an overall measure of the distance between the views.

An overview of the approach is presented in Figure 5.3. Object silhouettes are first represented by shock graphs (Transition 1). The shock graphs are represented in terms of shock trees using a minimum spanning tree of the weighted shock graphs and then embedded into normed vector spaces via spherical coding (Transition 2). This embedding technique ensures that distances between nodes map to Euclidean distances between their corresponding vectors, with low distortion. The distance between distributions is calculated using the Earth Mover's Distance under transformation (Transition 3). The details of this procedure are described in Demirci *et al.* [15].

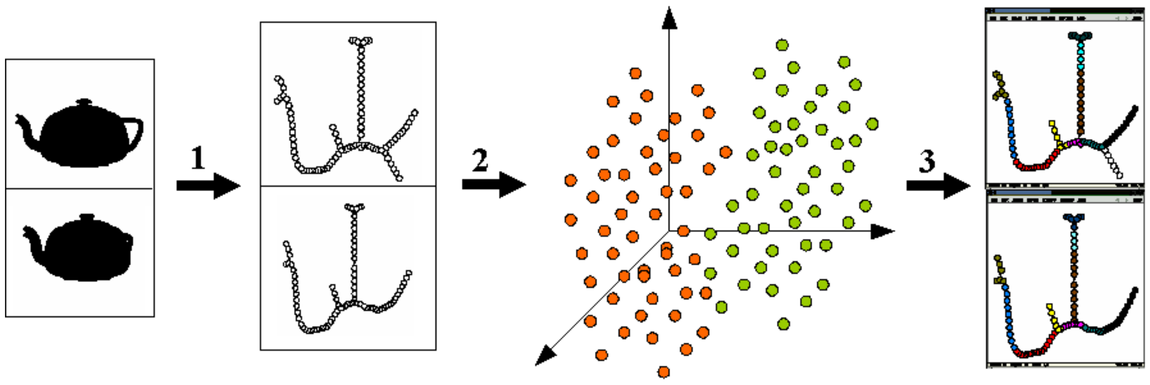


Figure 5.3: Illustration of the process used to compute the distance between two given views. Object silhouettes are first represented by shock graphs (Transition 1). The shock graphs are represented in terms of shock trees using a minimum spanning tree of the weighted shock graphs and then embedded into a normed vector spaces via spherical coding (Transition 2). The distance between distributions is calculated using the Earth Mover's Distance under transformation (Transition 3) [15].

5.3 Selecting Canonical Views Using MDMC

In the broader context of pattern simplification, computing "equivalent views" for pattern class $\mathcal{P}$ is closely related to that of the clustering problem. Namely, if the size k of the canonical set, the number of typical views, is more or less known, then one can use a clustering algorithm to partition $\mathcal{P}$ into subsets $\mathcal{P}_1, ..., \mathcal{P}_k$, and then choose an element c_i near the center of each cluster as the typical element of $\mathcal{P}_i$ for each $1 \leq i \leq k$. This technique has, in fact, been used to define aspect graphs for polyhedra [103, 107], and solids of revolution [20]. Clearly, if the number of subsets k in the partition of $\mathcal{P}$ is not known, then defining a centroid c_i based on a pair-wise similarity function will be extremely difficult. This concern provided motivation for the development and use of the MDMC approximation algorithm (3.25),

(MDMC): (5.1)

$$\text{Maximize} \quad \lambda_1 \left(\frac{1}{2} \sum_{i=1}^{n} (1 - y_i y_{n+1}) \right) + \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij} (1 - y_i y_j) \right), \tag{5.2}$$

$$\text{Subject to} \quad (1 + y_i y_{n+1}) + \sum_{j=1}^{n} A_{ij} (1 + y_j y_{n+1}) \geq 2, \; \forall\, 1 \leq i \leq n, \tag{5.3}$$

$$y_i \in \{-1, +1\}, \; \forall\, 1 \leq i \leq n+1, \tag{5.4}$$

where λ_1 and λ_2 are non-negative and $\lambda_1 + \lambda_2 = 1$.

In the context of 2D view selection, each 2D view will be represented by a vertex in a graph. The edges of the graph have weights corresponding the similarity between the 2D views. The MDMC algorithm, will select views with the following attributes:

1. Views in the canonical set are maximally similar to views not in the canonical set.
2. The size of the canonical set is as small as possible.
3. Every view is either in the canonical set or is similar to a view in the canonical set.

5.3.1 MDMC Experiments

This section presents an overview of experiments performed to evaluate the MDMC algorithm for computing canonical sets of 2D views. Each pattern class in the experiments corresponds to a set of 2D views acquired from a 3D object. Specifically, 9 objects are used, each representing a single pattern class. Each object has as many as 180 2D views acquired along a great circle of the viewing sphere, giving a total of 1620 views. A representative view of each object is shown in Figure 5.4.

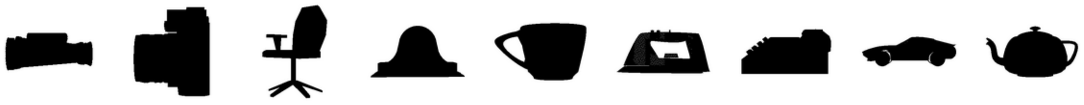


Figure 5.4: Sample views of the 3D objects used in the MDMC experiments

To compute the similarity values between 2D silhouettes corresponding to each 3D object (pattern class), the distance between them is calculated using the techniques described in Section 5.2. This procedure produces a distance matrix for each pattern class. To form similarity matrices, and thus the graph $G(\mathcal{P})$, the distance between all elements whose distance is greater than the mean distance is set to zero and a similarity value of e^{-d} is used for distances d less than the mean. Thus the similarities between views are thresholded to produce a graph which is not complete. The threshold indirectly influences the size of the resultant subset, *i.e.* a complete graph would result in a subset with one element.

Choosing the parameters λ_1, λ_2 is a critical step in generating the combined objective function. Note that the two parameters may be expressed as $\lambda_1 = \alpha$ and $\lambda_2 = 1-\alpha$. To find a good α value, a set of experiments was performed in which the value of objective 5.2 was measured as the value of α was varied in the interval $[0, 1]$. It was observed that a balanced trade-off between the cardinality of the canonical set and the similarity objectives, *i.e.*, $\alpha = 0.50$, produced the most consistent results. This by no means is a general assertion, and certainly needs a closer investigation with respect to the similarity function.

Results of this experiment can be seen in Figures 5.1(b), 5.5, 5.6, and 5.7. Examining the results subjectively, the results shown in Figure 5.1(b) illustrates an excellent subset of views. The 2 views of the clock which were selected out of the input set of 68 are quite meaningful, including both frontal and profile views. The results shown in Figure 5.5 are also good and show frontal and partial side views of the Porsche that are highly informative. In Figure 5.6, the two views of the camera in the fourth row are very similar, one of them is redundant. Also, note that the inclusion of a side view such as the one at the far bottom right would have been more informative. In Figure 5.7 the results are better. Even though two redundant views of the chair exist in the fifth row, both frontal and profile views are included. The inclusion of redundant views in the canonical set of 2D views may be a result of the rounding and approximation process.

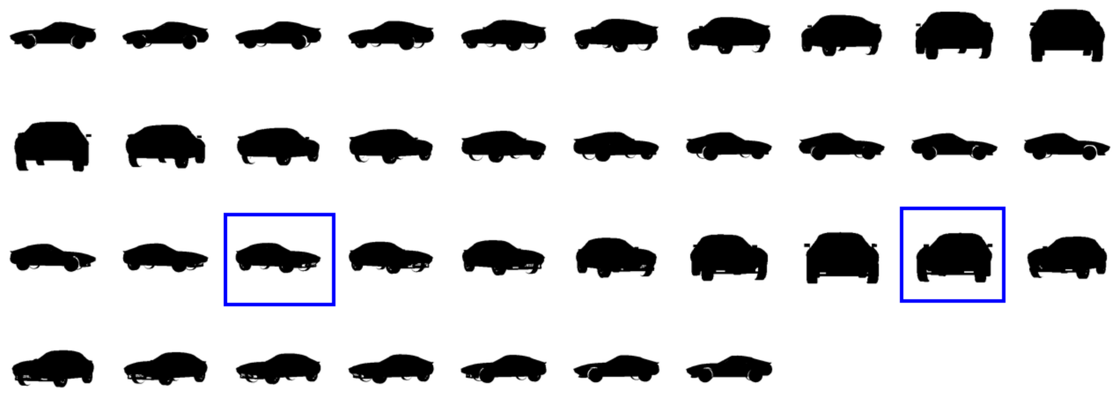


Figure 5.5: Canonical subset (blue rectangles) of 37 views of Porsche created using MDMC algorithm.

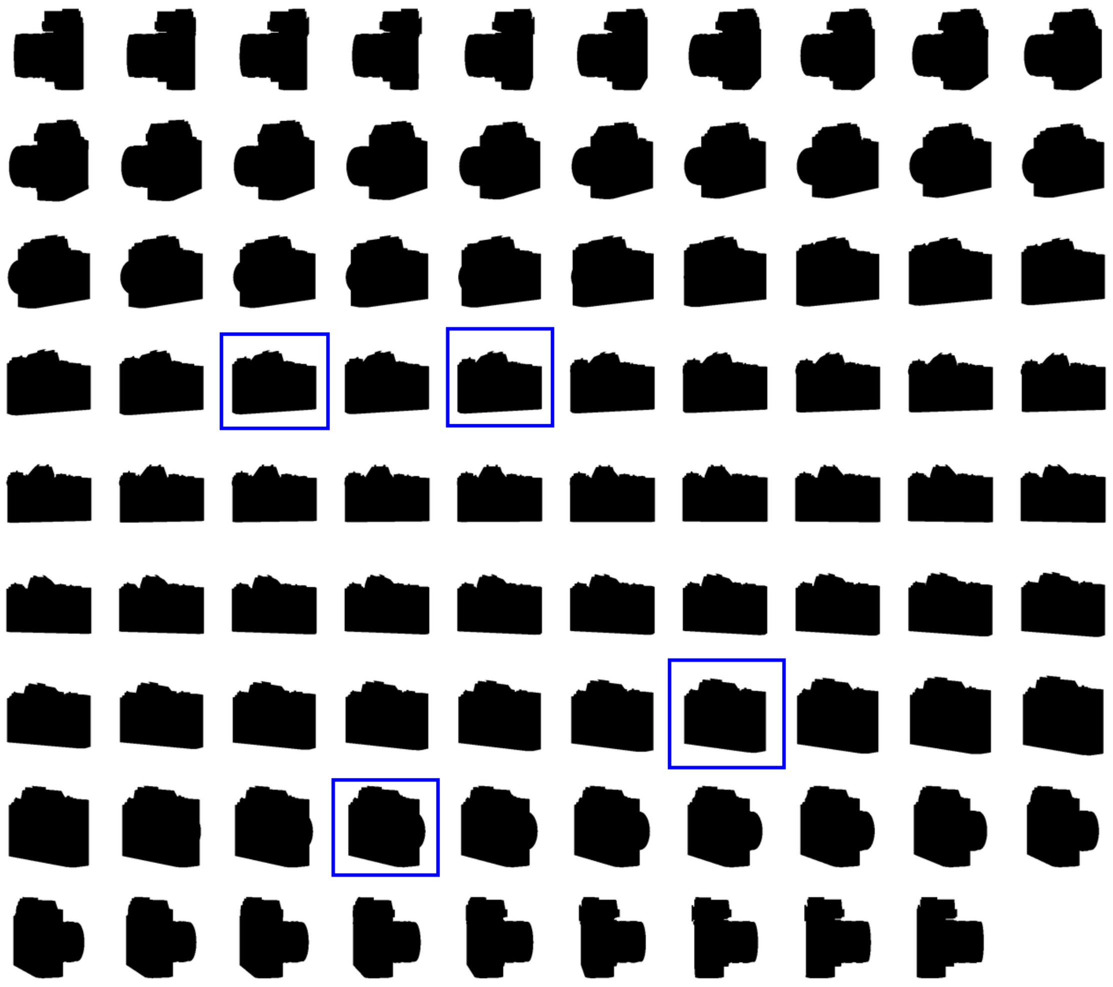


Figure 5.6: Canonical subset (blue rectangles) of 89 views of camera created using MDMC algorithm.

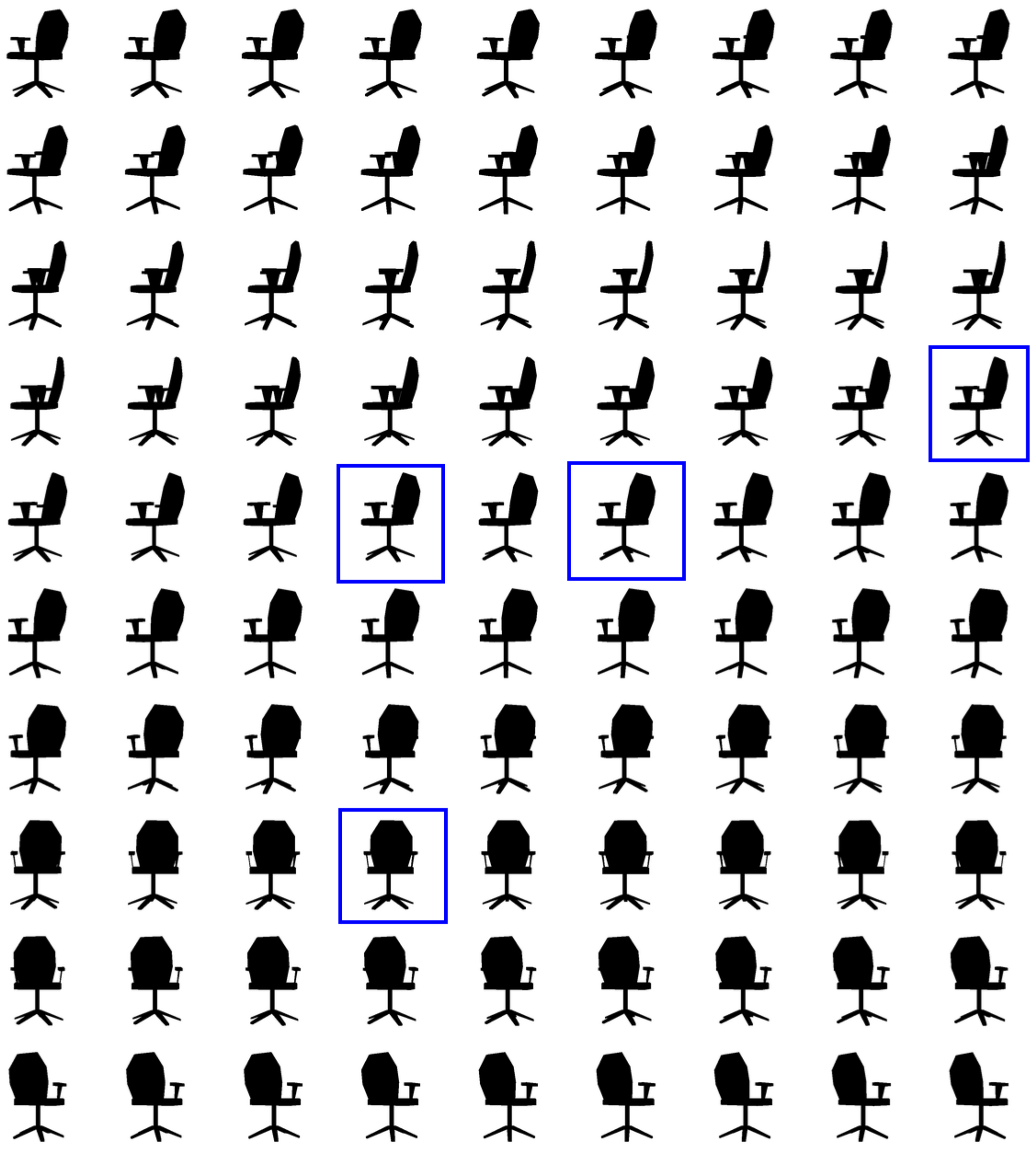


Figure 5.7: Canonical subset (blue rectangles) of 90 views of chair created using MDMC algorithm.

Next 9 smaller pattern classes were created from the original classes, with about 20 views per object. For each pattern class, an exhaustive search was conducted over the space of all possible subsets for the canonical set with the optimal objective value. Then the MDMC algorithm was used to compute an approximate solution. Figure 5.8 shows the performance ratio between the objective values of the canonical set obtained from the MDMC algorithm with that of the exhaustive search. In these cases, results of algorithm (3.25) were within a factor of 0.939 of exhaustive search. This result confirms the theoretic bound for the MDMC algorithm established in Section 3.3.6.

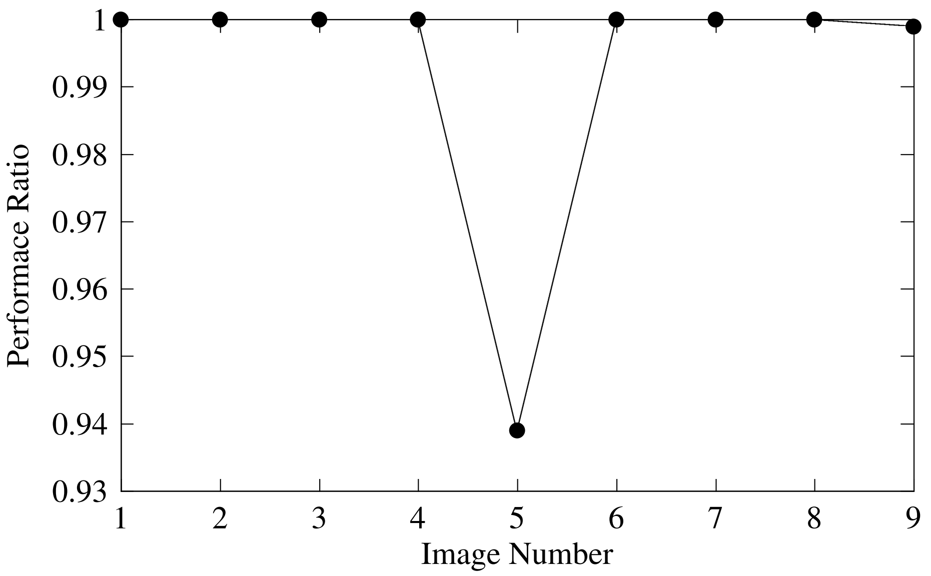


Figure 5.8: Comparing the similarity objective for canonical sets obtained from the MDMC algorithm (3.25) and for exhaustive search.

Figure 5.9 illustrates the sample views of several pattern classes and the canonical sets computed for each class (outlined in blue). The optimal sets found by exhaustive search are indicated in red.

Subjectively examining the results in Figure 5.9, it can be seen that in the case of the

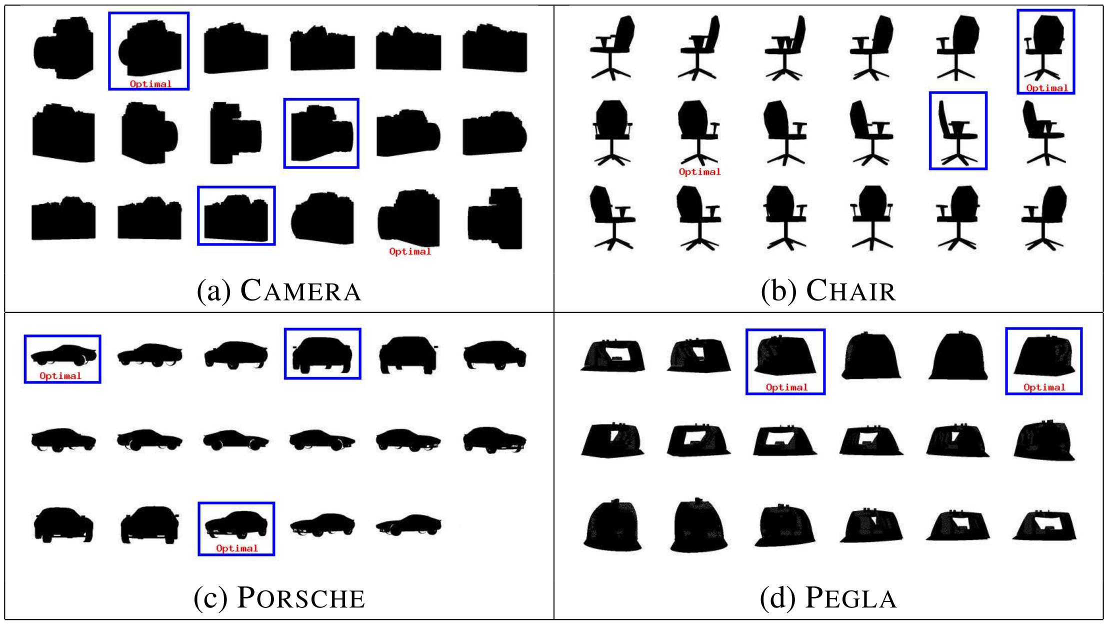


Figure 5.9: Canonical sets (blue rectangles are MDMC) for four sets of views. The optimal sets found by exhaustive search are indicated in red.

Camera (a), the optimal view in the last row is a rotation of the MDMC view in the second row. In fact the MDMC views in the second and third rows seem to make more sense than the 2 optimal views. In the case of the Chair (b) the 2 MDMC views seem better than the optimal set. The remaining images (c) and (d) show a close correspondence between the optimal and MDMC subsets.

Inclusion of views that were similar, as in the case of Figures 5.6 and 5.7, provided motivation for the inclusion of an objective term that would minimize the similarity of the views that were included in the canonical subset.

5.4 Selecting Views Using BCS

The experimental results described in Section 5.3.1 provided motivation to reformulate the canonical set problem with the goal of generating subsets where the members of the canonical subset are as dissimilar as possible. This property is useful when dealing with patterns belonging to multiple objects and one desires canonical elements that best represent one object to be maximally dissimilar from representative elements of other objects. In addition, upper (k_{max}) and lower (k_{min}) bounds on the cardinality of the canonical set were incorporated. Such bounds are important for applications that required a representative set of prescribed size. Finally, the restrictive assumptions about the similarity function will be removed, *i.e.*, all elements are explicitly comparable. This formulation is known as the bounded canonical set (BCS)[18],

(BCS): (5.5)

$$\text{Maximize} \quad \lambda_1 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - y_i y_j) + \frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - y_i y_{n+1})(1 - y_j y_{n+1}) \right)$$

$$+ \lambda_2 \left(\frac{1}{4} \sum_{i,j} \mathcal{W}_{ij}(1 - y_i y_j) \right), \tag{5.6}$$

$$\text{Subject to} \quad \frac{1}{2} \sum_{i=1}^{n} (1 + y_i y_{n+1}) - k_{min} \geq 0, \tag{5.7}$$

$$k_{max} - \frac{1}{2} \sum_{i=1}^{n} (1 + y_i y_{n+1}) \geq 0, \tag{5.8}$$

$$y_i \in \{-1, +1\}, \; \forall \, 1 \leq i \leq n+1, \tag{5.9}$$

where λ_1 and λ_2 are non-negative and $\lambda_1 + \lambda_2 = 1$.

5.4.1 BCS Experiments

This section presents an overview of the experiments performed to evaluate the method for computing the BCS and its applicability to view selection for view-based 3D object recognition. Each pattern class in the experiment corresponds to a set of 2D views acquired from a 3D synthetic object. The database consists of 9 objects with 19 views each, for a total of 171 views. The 2D views are acquired by sampling the surface of a view sphere centered on the object. A representative view of each object is shown in Figure 5.4. In a view-based 3-D recognition framework, the goal is to select a small number of views for each object in order to minimize search complexity at recognition time (assuming, for example, that recognition is performed using a linear search of the resulting selection).

For each of the objects, similarity matrices are constructed using the reciprocal of the distance computed using the technique described in Section 5.2. Then the BCS algorithm is used to compute subsets of views of each object type. The canonical sets are then combined to form a summary set, while the remaining views of each object are used as query views. Using the matching algorithm described in Section 5.2, each query view is compared to each element in the summary set. For each query view, the elements of the summary set are ranked in decreasing order of similarity. Then the position of the query's nearest canonical view (in the query's object's BCS) was recorded.

If one of the top six (closest) views in the summary set belonged to the same object, the match is considered a good match. Figure 5.10 shows how the matching rate changes as the size of the summary set is varied by changing the k_{min} and k_{max} parameters for the BCS algorithm.

The correct canonical view was found to be among the top 6 elements in the ranking 90.6% of the time when the size of the summary set was 48 views. The correct canonical view for a nearby query will not be top-ranked if there is another element of the summary set which is closer. This may happen when different objects share similar views which, in turn, may yield a summary set with similar elements. There is also an important trade-off between search complexity (size of the BCS bounds) and recognition accuracy (rank of the correct response). If, for example, the bounds are set too low given the complexity of the object, then there will be whole classes of object views that are not represented in the object's BCS. Thus, even though each query view on the object's view sphere will have a closest view in the BCS, that closest view may not be "nearby", thereby increasing the probability that an arbitrary member of another object's BCS (in the summary set) will be

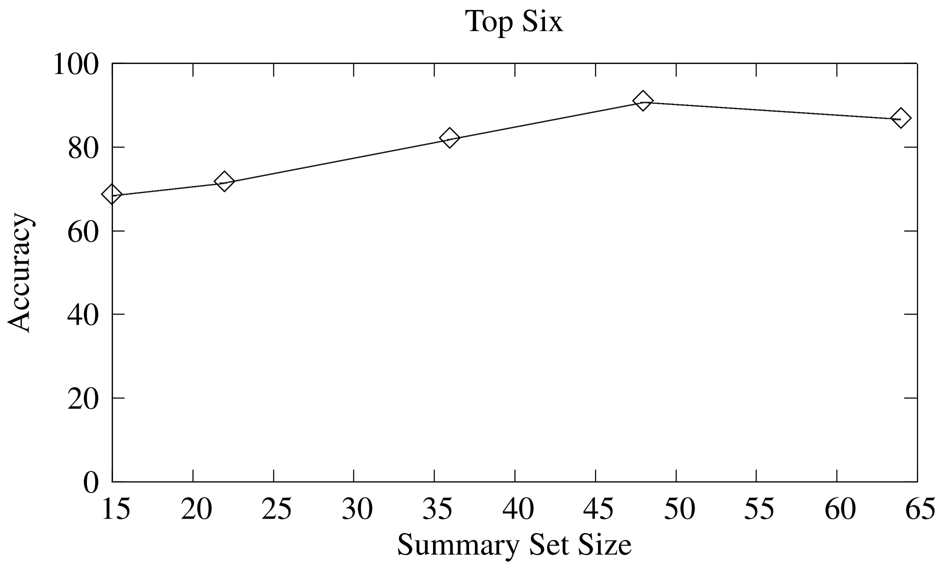


Figure 5.10: Matching results for top 6 as size of summary set is varied

closer to the query. The larger an object's canonical set, the greater the model coverage, the better the recognition accuracy, and the greater the search complexity.

Figure 5.11 shows some of the canonical sets computed by the BCS algorithm. In comparison to the canonical sets found using the MDMC algorithm shown in Figure 5.9, there is less redundancy. For example in the case of Figure 5.9(d), none of the views depicting the hole in the handle were selected, while in Figure 5.11(d) two of the selected views do. This can also be seen by comparing Figures 5.7 and 5.11(a), and also Figures 5.6 and5.11(b).

5.5 Conclusion

This chapter has presented results from a set of experiments showing the algorithms presented in Chapter 3 are useful for the task of 2D view selection and view-based 3D object recognition. First, in Section 5.2, a distance measure was described that can be used to compute the pair-wise similarity between 2D views. Section 5.3 then described the view

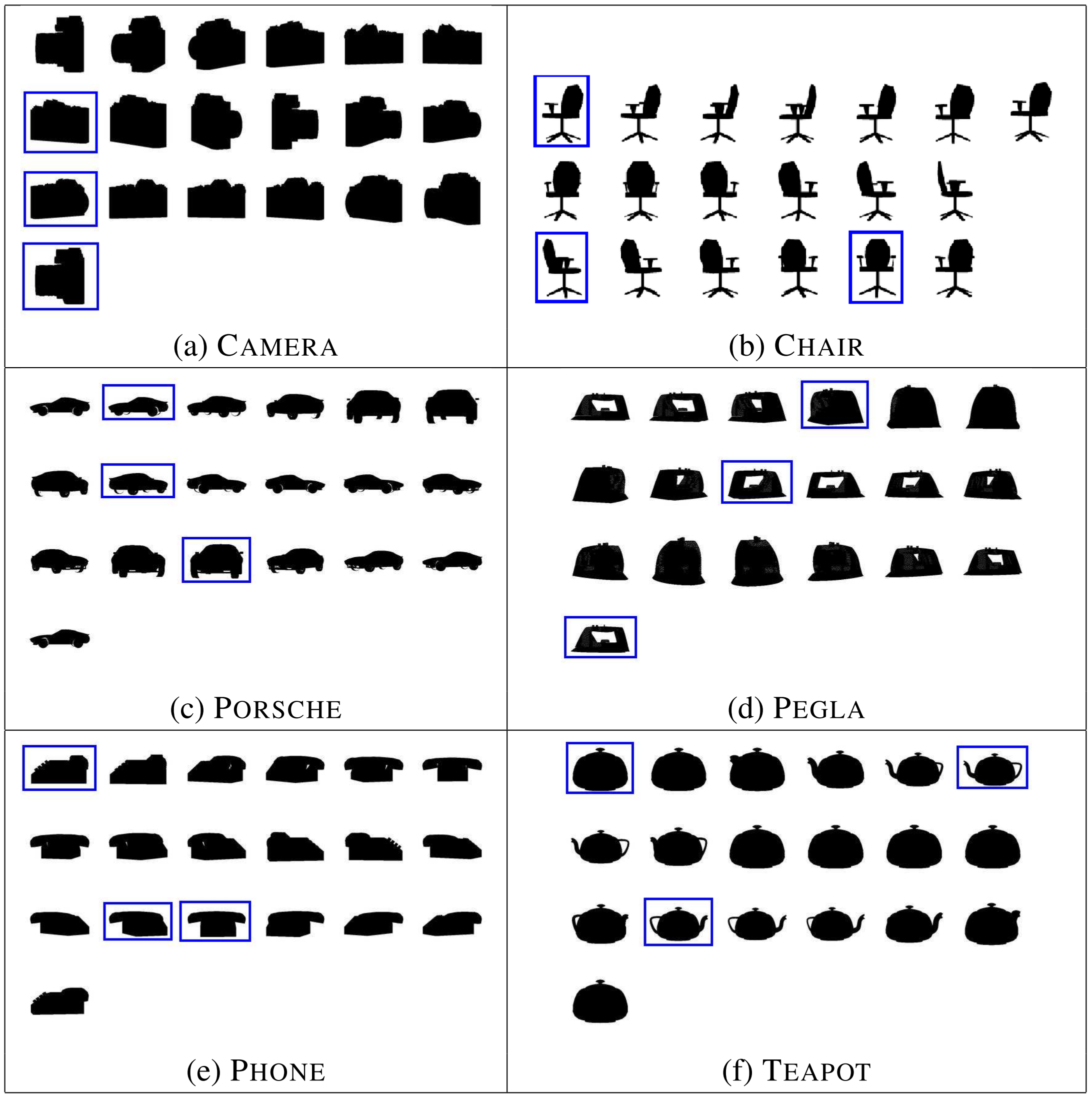


Figure 5.11: BCS (blue rectangles) for six sets of views.

selection technique and presented results from experiments using the minimum dominating max cut (MDMC) canonical subset algorithm. Finally, Section 5.4 presented results from experiments using the bounded canonical set (BCS) algorithm for view-based 3D object recognition.

6. Object Recognition

6.1 Introduction

Feature extraction followed by correspondence matching is fundamental and integral to many applications in computer vision including stereo, tracking, and object localization. In the context of computer vision the term *feature* often has two different meanings. It could mean a local object in an image such as an edge, or it could mean a point in a statistical pattern space. The difference between these meanings has been increasingly blurred in light of recent developments in feature detection [75, 50], and here the term feature is used to describe both of these phenomena.

Given a set of features in a query image and a set of features in a target image, correspondence matching may be defined as the process of determining which target features represent which query features. The goal is to establish one-to-one relationships between features in the query and target sets. Increasing the difficulty, is that the two images may not be exact copies, as one may have undergone some transformation. Once the correspondence is known, the transformation between the images may be calculated.

A method of solving similarity transformations using aligning correspondence points was originally described by Huttenlocher and Ullman [40, 41]. They assumed that three corresponding points were known between a query and a target, and presented an algorithm to solve for the translation, scaling, and rotation. The algorithm has since been applied to a number of applications including randomized point matching [43], object recognition [95], and shape tracking [80].

Early methods of image matching using local features stemmed from Harris's corner and edge detector [36]. The corners and edges detected using Harris's method depended on the current scale of the image. Matching techniques using the features were unable to differentiate two images of the same object residing at different scales. Further work has been done on representing images in scale-space, where images are represented by a set of features residing at a number of different scales. Lindeberg has worked on the problem of extracting features in scale-space with automatic scale selection [69]. More recently, Lowe introduced a method named scale invariant feature transform (SIFT), which extracts local scale-space features which include a number of scale-space characteristics [75]. Vectors containing the feature's location in scale-space as well as scale-space characteristics are used for image matching.

Recent trends in feature detection have been in the development of methods that can extract large numbers of rich features from a given image. One consequence of these trends has been the increased size of an object's representation.

The size of an object representation, or memory footprint, is a combination of the number of features and the feature descriptor size. The memory footprint of an object's representation can be made smaller by reducing the size of the feature descriptor, by reducing the number of features, or both. Recent work by Ke and Sukthankar [51] has shown that feature descriptors can be reduced in size. The focus here, which is complementary to theirs, is on reducing the number of features. The theory is that some of the detected features are more representative than others, and a subset of detected features is sufficient for vision tasks like object localization.

The fact that many feature detectors include additional information about the feature besides the feature's location provided motivation for the development of the SBCS algorithm described in Section 2.7.3. The idea is that a notion of stability can be used to influence the selection of canonical features. The stability of a feature is a quantitative measure of its response to the feature detector.

Recall that the SBCS approximation algorithm (3.27) is

$$\begin{aligned}
&\textbf{(SBCS):} && \\
&\text{Maximize} && \mathcal{C} \bullet \mathcal{X} \\
&\text{Subject to} && \mathcal{D}_i \bullet \mathcal{X} \geq 0,\ \forall\, i = 1\ldots, m, \\
& && \mathcal{X} \succeq 0,
\end{aligned}$$

where

$$\mathcal{C} \;=\; \lambda_1 \begin{bmatrix} \tilde{\mathbf{0}} & -\frac{1}{4}\mathbf{d} \\ -\frac{1}{4}\mathbf{d}^T & \frac{1}{2}w_\Sigma \end{bmatrix} + \lambda_2 \begin{bmatrix} -\frac{1}{4}\mathcal{W} & \hat{\mathbf{0}} \\ \hat{\mathbf{0}}^T & \frac{1}{4}w_\Sigma \end{bmatrix} + \lambda_3 \begin{bmatrix} \tilde{\mathbf{0}} & -\frac{1}{4}\mathbf{t} \\ -\frac{1}{4}\mathbf{t}^T & \frac{1}{2}t_\Sigma \end{bmatrix},$$

and $\tilde{\mathbf{0}}$ is an $n \times n$ matrix of zeros, $\hat{\mathbf{0}}$ is an all zeros column vector in $\mathbb{R}^n$, $\mathbf{d}$ is a column vector in $\mathbb{R}^n$ whose i^{th} entry has value $d_i = \sum_{j=1}^{n} \mathcal{W}_{ij}$, $\mathcal{W}$ is the $n \times n$ similarity matrix, $w_\Sigma = \sum_{i,j} \mathcal{W}_{ij}$, $\mathbf{t}$ is a column vector in $\mathbb{R}^n$ whose i^{th} entry is t_i, and $t_\Sigma = \sum_{i=1}^{n} t_i$. Constraint matrices $\mathcal{D}_1, \ldots, \mathcal{D}_{n+1}$ are as described in Eqn. (3.22), $\mathcal{D}_{n+2}$ is as described in Eqn. (3.23), $\mathcal{D}_{n+3}$ is as described in Eqn. (3.24), and $m = n + 3$. The coefficients, $\lambda_1 \geq 0, \lambda_2 \geq 0$, and $\lambda_3 \geq 0$ are parameters such that $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

That is the SBCS algorithm seeks to identify a canonical subset, $\mathcal{P}^*$ with the following attributes:

1. Features in the canonical set are minimally similar.
2. Features in the canonical set are maximally similar to features not in the canonical set.
3. Features in the canonical set are maximally stable.
4. The size of the canonical set is as least k_{min} and at most k_{max}.

The experiments presented in this chapter explore how a subset of image features created using the SBCS algorithm may be used to perform object localization. Figure 6.1 shows an overview of the technique. Two images, one a transformed copy of the other, are represented as a large number of features. Using the SBCS algorithm, a subset of features is extracted, the transformation between the query and target image is determined, and then the results are visualized. In the case shown in Figure 6.1, the transformation aligns the images with negligible error.

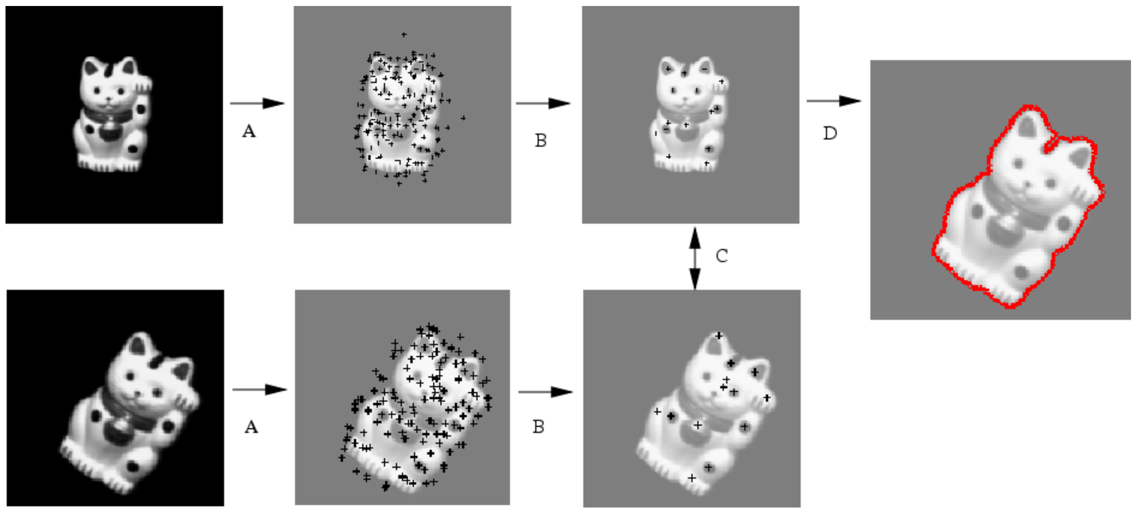


Figure 6.1: A) Features are extracted from query (top) and target (bottom). B) The SBCS algorithm is used to extract subsets of features. C) The transformation between query and target is determined. D) Outline shows transformation determined from SBCS.

Section 6.2 presents an overview of the feature detectors used in the experiments. Section 6.3 describes how the feature correspondences between query and target are computed. Then Section 6.4 discusses the similarity measures used in the experiments. Section 6.5 presents the stability measures used in the experiments. Section 6.6 presents experiments that show that a canonical subset of image features can be used to localize one object. Section 6.7 presents experiments with one object rotated in depth. Section 6.8 presents some results from experiments localizing multiple occluded objects. Then Section 6.9 presents results from experiments testing pose estimation with out-of-plane rotations. Next Section 6.10 presents some results of experiments localizing occluded objects in synthetic scenes along with a measure of the localization accuracy. Section 6.11 examines the relative computational cost of object localization as the size of a subset of features increases. Then Section 6.12 presents results of experiments that tests how the feature subsets created with the SBCS algorithm reacts with noise. This is followed by Section 6.13 where the SBCS subset selection method is compared with other commonly used techniques of computing feature subsets on the task of localization of occluded objects in synthetic scenes. Finally, in Section 6.14, some results are presented from experiments that test object localization under occlusion with images that are real.

6.2 Feature Detectors

Feature detection is the process of isolating *interesting* portions of an image that conform to some statistical measure for the purposes of understanding or analyzing the image. What is meant by *interesting* may be determined by the particular application. For instance in satellite surveillance, interesting features may correspond to parts of buildings, roads, or vehicles such as tanks. For facial recognition, perhaps one is more interested in the location of the eyes. Image features may be considered as fundamental units of meaning that may be combined together through other processes such as segmentation to produce a deeper understanding of the image. The intent is to discover structure within an image. As such, feature detection can be thought of as a pre-processing step that seeks to model the image at a low level. In general, the features are then used in subsequent tasks such as segmentation or recognition rather than the original pixel data.

The following three feature detectors will be used in the object recognition experiments presented in this chapter.

1. HARRIS is the corner and edge detector described by Harris [36]. The detector returns a single scalar at each pixel location that indicates the response of the detector at that point.

2. BLOB-RIDGE is the scale-space feature detector introduced by Lindeberg and Bretzner [7, 69, 70] that detects blobs and ridges. The detector returns the position in x, y coordinates of the features along with their scale, significance, orientation and whether the feature is a blob or a ridge. The scale of the feature is indicative of the support region around the feature location that influences the feature. The significance of a feature is a measure of the response of the detector that is normalized to compensate for differences in scale. Figure 6.2 shows an image marked with blobs and ridges detected in scale-space. In order to simplify the presentation of the features, while visualizing as many features as possible, henceforth only the centers of the blobs and ridges will be shown.

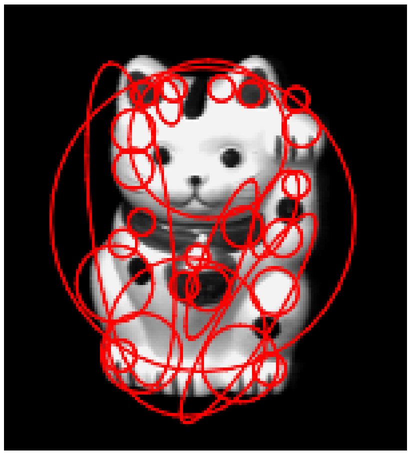


Figure 6.2: Blobs and ridges detected in scale-space

3. SIFT is the scale invariant feature transform introduced by David Lowe [75]. The

detector returns the x, y coordinates of the feature along with the scale, orientation, and a 128 byte descriptor vector. The descriptor vector is an oriented histogram of thresholded gradients which describes the feature's immediate neighborhood in scale-space. Figure 6.3 shows an image marked with SIFT features showing orientation direction and scale. In order to simplify the presentation of the features, while visualizing as many features as possible, henceforth only the locations of the SIFT features will be shown.

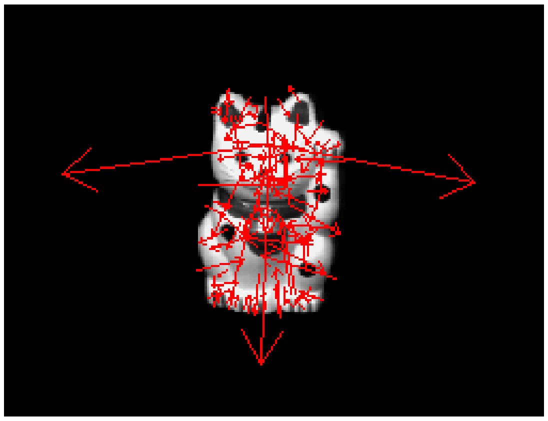


Figure 6.3: SIFT features (red) showing orientation direction and scale

6.3 Object Localization

Object localization is the process of determining the position, orientation, and scale of a query object (model) in a target scene. The goal is to find a transformation, T, that can be applied to the query such that it can be superimposed on the target scene in the correct location, with the proper scale and orientation. To calculate T, corresponding features must be found, *i.e.*, features in the query must be matched to features in the target. Figure 6.4 shows an illustration with correspondences between features in a query and target image.

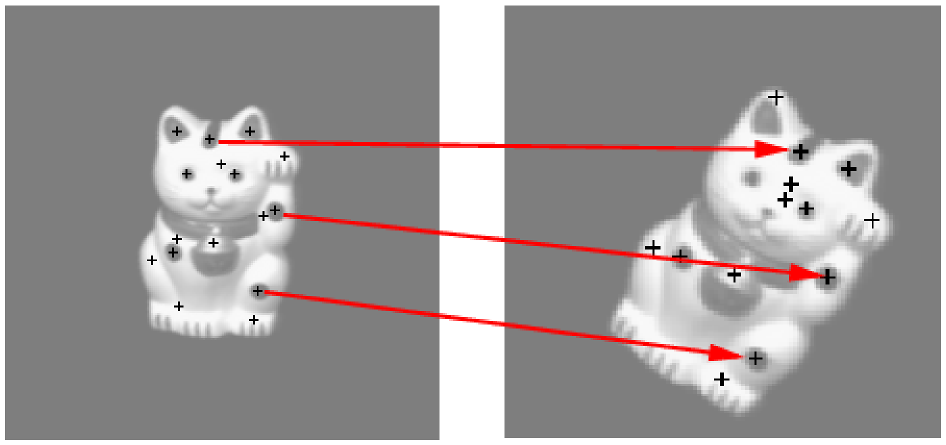


Figure 6.4: Correspondence between features

The experiments described in this chapter were conducted using two matching algorithms. The first, which is denoted HUT-EMD, is based on Huttenlocher and Ullman's algorithm [40, 41]. HUT-EMD finds a transformation, T, for each possible set of three query features and three target features in the reduced feature subsets. Each transformation is ranked by applying it to the full feature set, and then measuring the distance between the transformed query and object features using a many-to-many matching algorithm [14] based on the Earth Mover's Distance (EMD). The transformation, T, resulting in the minimum distance is considered the best.

The second matching algorithm, which is denoted MATCH, was used exclusively for matching SIFT features. MATCH is supplied as part of the SIFT package [74]. This algorithm takes each query feature and finds the two nearest neighbors in the target feature descriptor space. If the distance to the nearest neighbor is less than a threshold percentage of the distance to the second nearest neighbor, the nearest neighbor is considered a match. Using pairs of corresponding features from the query and target images supplied by the MATCH algorithm, standard linear algebra techniques may be used to calculate T.

6.4 Similarity

Two similarity measures between features are used in the experiments presented in this chapter. The first, which is denoted XY, is defined as the inverse of the Euclidean distance between two feature locations in the image. This measure of similarity is designed to distributed the selected features spatially across the image.

The second similarity measure, DSPACE, is defined as the inverse of the Euclidean distance between features in the descriptor space, and will only be used with SIFT features. This similarity measure is designed to distribute the selected features spatially across the high dimensional feature descriptor space.

6.5 Stability Measures

The SBCS algorithm uses a stability assigned to each feature in conjunction with a similarity measure. The stability measures used in the experiments described in this chapter are all well-defined with the exception of PERTURB, which is described in Section 6.5.1.

1. RES is a stability measure that is used with the HARRIS detector, and is defined as the scalar response of the detector.
2. SIG is the significance field of the BLOB-RIDGE features supplied by the detector.
3. SCALE is a stability measure that is used with the SIFT detector. A SIFT feature includes the scale at which the feature is found in the input image. The scale of a feature is indicative of the support region, higher scales indicate larger support regions.
4. PERTURB is a stability measure that is computed, for a description see Section 6.5.1.
5. I-PERTURB is defined as $1/(1 + PERTURB)$.

6.5.1 PERTURB Measure

The PERTURB stability measure is computed by perturbing the query images, and then matching the unperturbed image against the perturbed image. The motivation for computing a stability value by perturbing the image comes from Lowe [75], where the stability of the SIFT features was tested under various transformations. Specifically, each of the query objects (see Figure 6.14) is subjected to the following perturbations:

- Contrast increased by a factor of 1.2.
- Intensity increased by a factor of 1.2.
- Rotation by 20 degrees.
- Scaled by a factor of 0.7.
- Scaled by a factor of 1.2 in the x direction.
- Scaled by a factor of 1.5 in the x direction.
- Addition of 10% Gaussian pixel noise.
- Combination of contrast increased by a factor of 1.2, intensity increased by a factor of 1.2, rotated by 20 degrees, scaled in x direction by a factor of 1.2, scaled in the y direction by a factor of 0.7, 10% pixel noise added.

SIFT features were then extracted from each of the 8 perturbed images. The MATCH algorithm (see Section 6.3) was then used to match features extracted from the unperturbed model images to features extracted from the corresponding perturbed model image. Then the stability is calculated of each feature as $\tau = e^d$ where d is the Euclidean distance between where the feature was detected in the perturbed image versus where it should have been. If a feature was not matched it has a stability value of zero. The final stability of the feature was the average stability over all 8 of the perturbations for that object.

6.6 Localization of One Object

In this experiment, the goal was to show that a SBCS of image features *could* be used to localize an object. Four objects were randomly chosen from the COIL-20 [86] database, and a random transformation was applied which included rotation in the image plane, scaling, and translation. Features were then extracted from the query (untransformed) images and the target (transformed) images using the BLOB-RIDGE feature detector described in Section 6.2. Then the SBCS algorithm was used to extract a subset of the image features. The inverse of the Euclidean distance between the feature locations was used as a similarity measure, and the significance field of the BLOB-RIDGE features was used as the stability. Finally, the HUT-EMD algorithm described in Section 6.3 was used to find and compute the transformation between the query and target images.

Figure 6.5 shows the results of this experiment. Column one of the figure shows the four images from the COIL-20 database and the canonical feature sets computed using the SBCS algorithm. The second column shows the same image rotated in the image plane and scaled, and the respective canonical set. In order to apply the HUT-EMD algorithm, three feature correspondences are needed, however it may be visually confirmed that the intersection of the canonical sets of the two images contain many more correspondences. The third column of the figure shows the outline of the first image transformed by the transformation found using the HUT-EMD algorithm. In each of the test cases the transformation computed is highly similar to the actual transformation.

6.7 Localization of One Object Rotated in Depth

In this experiment the same four objects from the COIL-20 database were used (see Section 6.6), and the same four objects rotated in depth five, ten, and fifteen degrees. Features were then extracted from the query (untransformed) images and the target (transformed) images using the BLOB-RIDGE feature detector described in Section 6.2. Then the SBCS

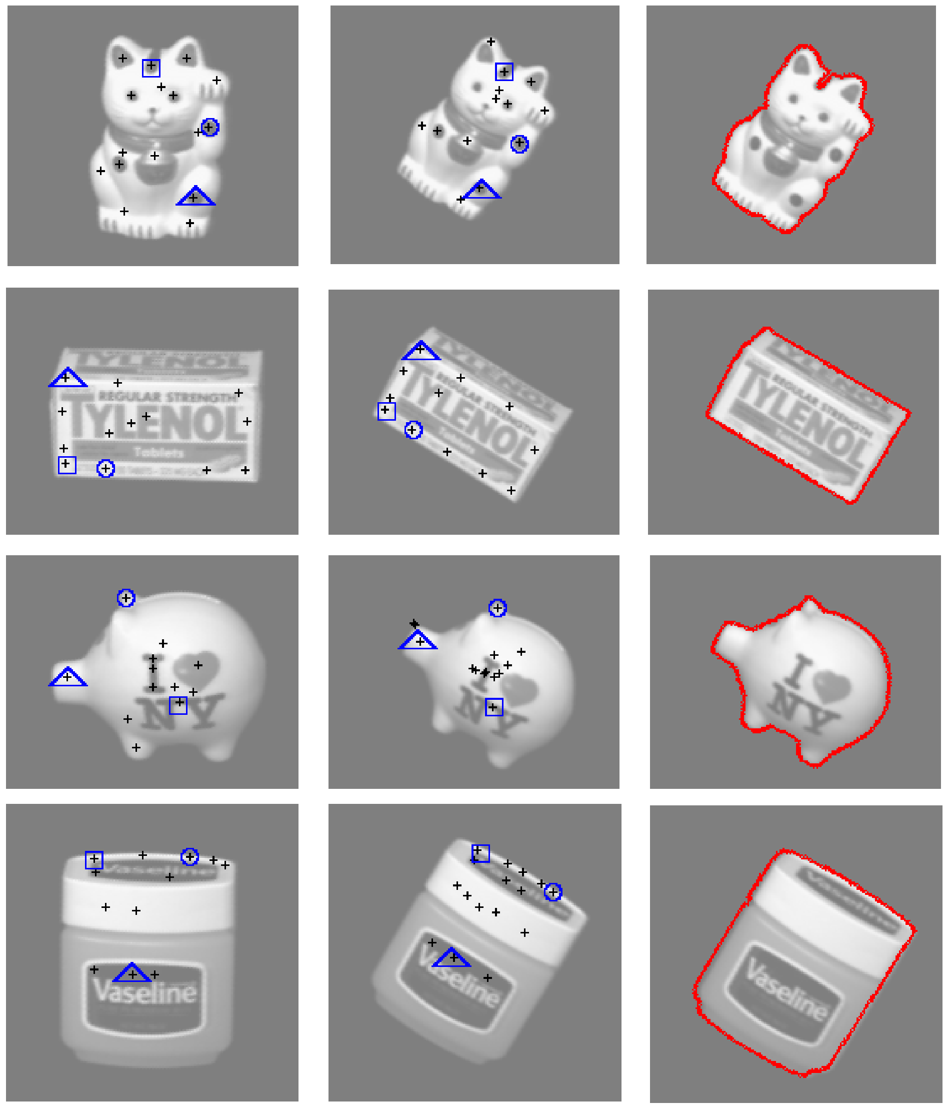


Figure 6.5: Left: reference images with SBCS features, Middle: images rotated and scaled with SBCS features, Right: outline of reference images transformed onto rotated images. The three corresponding features used to determine the optimal transformation are marked in blue with a circle, triangle and square.

algorithm was used to extract a subset of the image features. The inverse of the Euclidean distance between the feature locations was used as a similarity measure, and the significance field of the BLOB-RIDGE features was used as the stability. Finally, the HUT-EMD algorithm described in Section 6.3 was used to find and compute the transformation between the query and target images.

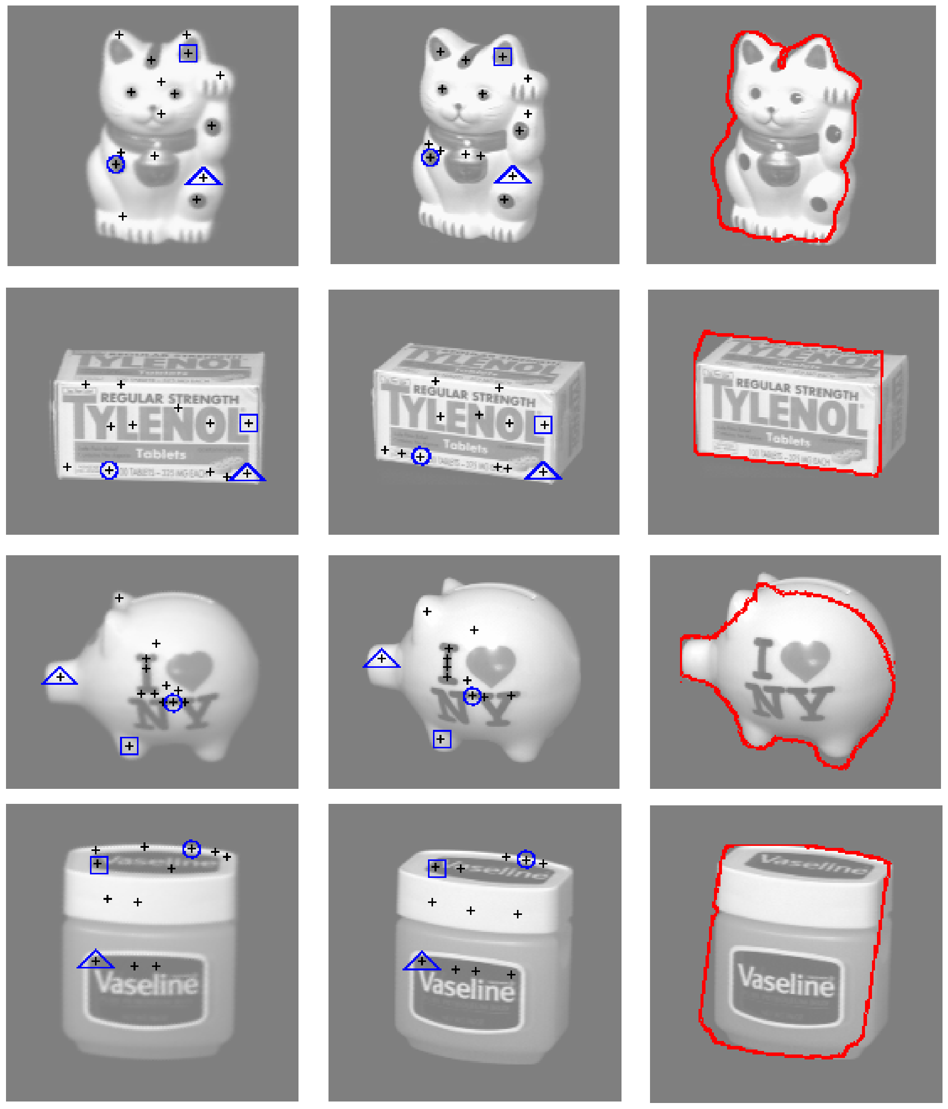


Figure 6.6: Left: reference images with SBCS features, Middle: images rotated in depth with SBCS features, Right: outline of reference images transformed onto rotated images. The three corresponding features used to determine the optimal transformation are marked in blue with a circle, triangle and square.

Figure 6.6 shows the results from this experiment for the fifteen degree rotations in depth (the other results were similar). The results show the invariance of the canonical subsets under small rotations in depth. The out-of-plane rotation experiments show that the method does not rely on seeing exactly the same view as the query view and can establish correspondence among stable features obtained by SBCS. Column one of Figure 6.6 shows the four query images from the COIL-20 database and the canonical feature sets computed using the SBCS algorithm. Column two shows the canonical features found in the same four objects rotated in depth fifteen degrees. Again the intersection of these canonical sets can be visually be confirmed to be greater than three features, the number required to compute the transformation. Column three shows the outline of the original image transformed by the transformation which results in the minimum distance between the original image and its copy rotated in depth. In each test case the transformation is not perfectly correct, as the technique computes a two-dimensional rotation matrix from the points in the feature space, while the target image has been rotated in depth by a three-dimensional rotation. Still, the computed transformation serves as a good approximation.

6.8 Localization of Multiple Occluded Objects

In this experiment the four objects used in the experiments described in Sections 6.6 and 6.7 were composited into a synthetic scene such that three of the objects were partially occluded. BLOB-RIDGE features were then extracted from the composite scene, and the SBCS algorithm was used to extract a subset of the image features. The inverse of the Euclidean distance between the feature locations was used as a similarity measure, and the significance field of the BLOB-RIDGE features was used as the stability.

The canonical sets of image features of the individual query objects, described in Sections 6.6 and 6.7, were then used as queries against the composite image. As before, the HUT-EMD algorithm described in Section 6.3 was used to find and compute the transformation between the query and target images. Figure 6.7 shows sample results of this experiment. Column one shows the composite image with several hundred BLOB-RIDGE features. Column two shows the fifty-five points of the SBCS computed from the features. Column three shows the outlines of the best transformations found for the four objects.

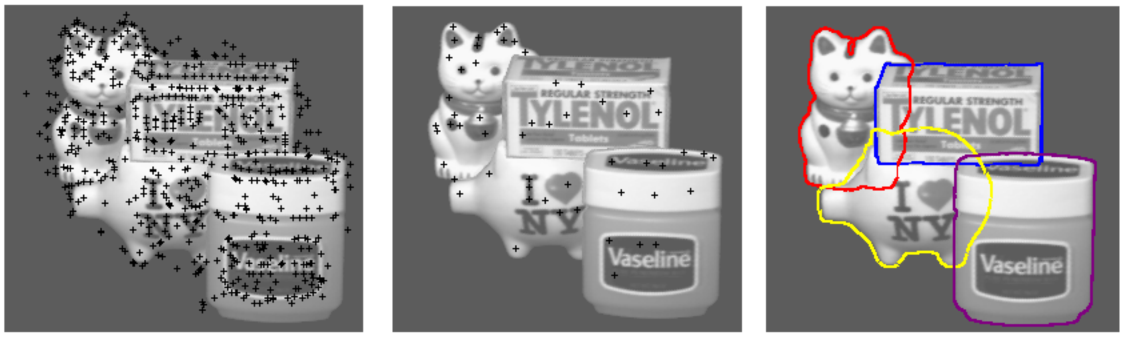


Figure 6.7: Left: reference image with scale-space features, Middle: image with SBCS features, Right: outline of query images on reference image.

6.9 Pose Estimation

This experiment tested pose estimation with out-of-plane rotations. The COIL-20 [86] database was used, which consists of images of 20 objects taken from 72 different viewing directions at 5 degree intervals about the vertical axis. The views of each object are numbered from 0 to 71. The odd views of each of the 20 objects were used as queries, and the even views were used as targets.

BLOB-RIDGE features were then extracted from the images, and the SBCS algorithm was used to extract a subset of the image features. The inverse of the Euclidean distance between the feature locations was used as a similarity measure, and the significance field of the BLOB-RIDGE features was used as the stability.

The queries were then matched against the even views of the same object and frontal views of the other 19 objects using the HUT-EMD algorithm described in Section 6.3. A query was considered to be successful if the image with the minimum distance to the query is a neighboring view of the same object – the object rotated 5 degrees in either direction. In total, we performed 720 queries of which 89.7% were successful. These results are comparable to a recent study conducted by Demirci *et al.* [14].

On average, each view was represented by 119 features, which was reduced using the SBCS algorithm to an average size of 12. By using the SBCS for pose estimation, the number of correspondences to be evaluated is reduced by a factor of six orders of magnitude *e.g.* from $\binom{119}{3}\binom{119}{3}$ to $\binom{12}{3}\binom{12}{3}$. These results indicate that the use of the SBCS results in a large reduction in the number of computations without significantly degrading the functional performance of the application.

6.10 Localization in Occluded Scenes

The results of the experiments in Sections 6.6 and 6.8 show that high quality localization can be obtained using a subset of image features. Nevertheless, evaluation of the quality of localization is subjective. One can examine the transformed outlines in Figures 6.5 and 6.7 and ascertain that the alignment looks good, but how good is it?

To answer this question, a database of synthetically occluded objects, DOOR [16], using the COIL-20 [86] image database was created. Each image contains multiple objects with occlusion. A text file accompanies each image containing the names of the objects, the amount of occlusion of each object, and their locations in the image. Having the ground truth information used to construct the occluded scenes enables the automated evaluation of localization results. Figure 6.8 shows five images with a varying number of objects and degrees of occlusion from the database.

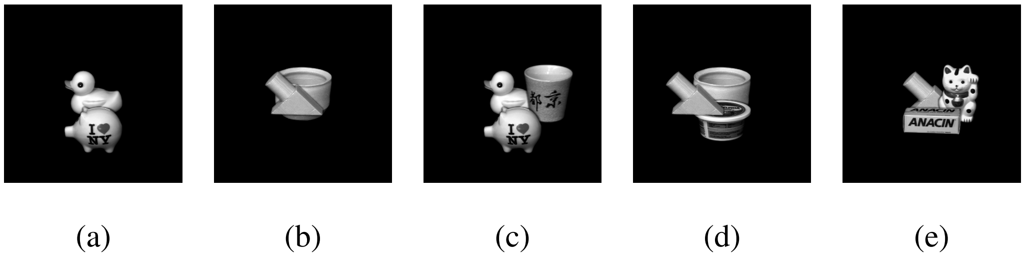


Figure 6.8: Example images from the DOOR dataset used in the experiments on object localization with partial occlusion. The amount of occlusion from left to right: (a) 18.85% (b) 56.04% (c) duck 33.73%, cup 3.51% (d) bowl 43.24%, cream cheese 23.90% (e) block 43.49%, cat 16.12%.

One way of measuring the accuracy of localization when the ground truth is known is to transform the object using the computed transformation T and also with the ground truth transformation T_G. The accuracy of the localization, denoted by Γ, can be computed from the area of the overlap between the target and the transformed query objects. The accuracy, Γ, is defined as the minimum of the normalized overlapping area with respect to both the target and the query object regions. Computing this bidirectional accuracy and choosing the minimum prevents high scores being given to degenerate cases such as scaling the query to subsume the target object. In a perfect localization $\Gamma = 1.0$. Figure 6.9 shows a visualization of the overlap areas. The red and green area shows the object silhouette transformed by the ground truth T_G. The yellow and green area shows the object silhouette transformed by the computed transformation T. The minimum normalized overlap $\Gamma = 0.462$.

In this experiment 660 images from DOOR were used, each with two or three objects with varying occlusion up to 60%. For each of these images, localization was attempted on each of the occluded objects in the scene. As in the experiments of Sections 6.6 and 6.8,

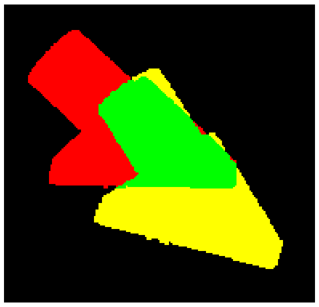


Figure 6.9: The red and green area shows the object silhouette transformed by the ground truth. The yellow and green area shows the object silhouette transformed by the computed transformation. The minimum normalized overlap $\Gamma = 0.462$.

features were extracted from the query images and the target occlusion scenes using the BLOB-RIDGE feature detector described in Section 6.2. Then the SBCS algorithm was used to extract a subset of the image features. The inverse of the Euclidean distance between the feature locations was used as a similarity measure, and the significance field of the BLOB-RIDGE features was used as the stability. Finally, the HUT-EMD algorithm described in Section 6.3 was used to find and compute the transformation between the query and target images.

Figure 6.10 shows example cases that resulted in high localization accuracy. For each of the examples the three corresponding points used to compute the transformation are labeled in the query and the target images. Figure 6.11 (a) and (b) show histograms of Γ for target images containing two objects and three objects, respectively. The histograms indicate the majority of the queries had a Γ value over 0.80, with 79% of queries for scenes containing two objects and 65% of queries for scenes containing three objects.

Figure 6.12 shows the localization accuracy under increasing amounts of occlusion for all target images. The average values of Γ are plotted for six discrete ranges of occlusion percentage. For scenes with two and three objects, Γ is always over 0.80 and 0.70, respectively. The decrease in the localization accuracy for images with three objects may be attributed to the increased complexity of the scenes (they contain more objects). Figure 6.13

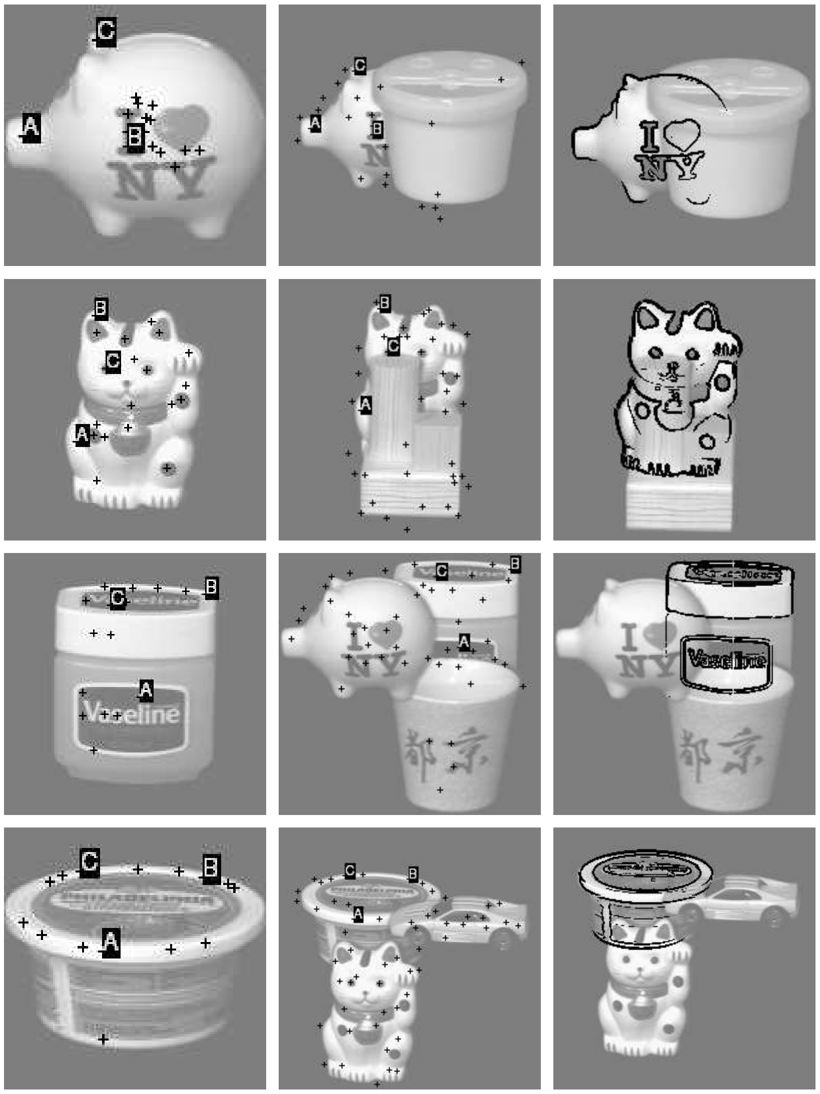


Figure 6.10: Example results from the experiment of localizing partially occluded objects. Left and middle columns show the query object images and the target images, respectively. In these images the features in the SBCS are depicted with crosses. The three corresponding features used to determine the transformation are labeled ‘A’, ‘B’, and ‘C’. The right column shows the edges of the localized query objects overlaid on the scene. (All images are histogram scaled and cropped for clarity.) Row 1: pig is 58.7% occluded, Γ=1.00, Row 2: cat is 58.7% occluded, Γ=0.97, Row 3: vaseline is 54.1% occluded, Γ=0.98, Row 4: cream cheese is 39.79 % occluded, Γ=0.96.

shows an example of a failed query. As this example depicts, failures can be caused by lack of discriminating texture in the unoccluded regions of the target objects.

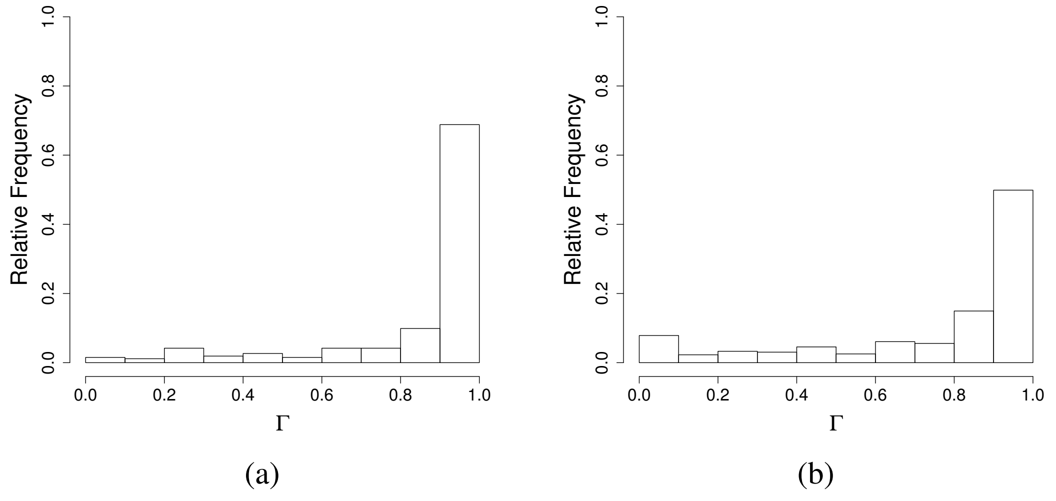


Figure 6.11: Results of the localization of partially occluded objects in scenes containing (a) two objects and (b) three objects. The histogram plots the relative frequency ratio of queries with varying localization accuracies, Γ. The histograms indicate the majority of the queries had a Γ value over 0.80 with 79% of queries for scenes containing two objects and 65% of queries for scenes containing three objects.

These results demonstrate that the SBCS successfully chooses representative features, even in the presence of significant occlusion. This is mainly because the SBCS is encouraged to maintain the spatial distribution of the original features. This increases the probability of having features in the query SBCS located in the corresponding unoccluded region of the target object.

6.11 Computational Cost of Subset Size with HUT-EMD

In this experiment, the goal was to determine the relative computational cost of object localization as the size of a subset of features increases. First a small dataset was created consisting of 20 query images from the frontal views of objects from the COIL-20 [86] dataset (see Figure 6.14). Each image was then randomly scaled and rotated in the image plane to form 20 target images.

Sets of 50 BLOB-RIDGE features were then extracted from the images, and the SBCS

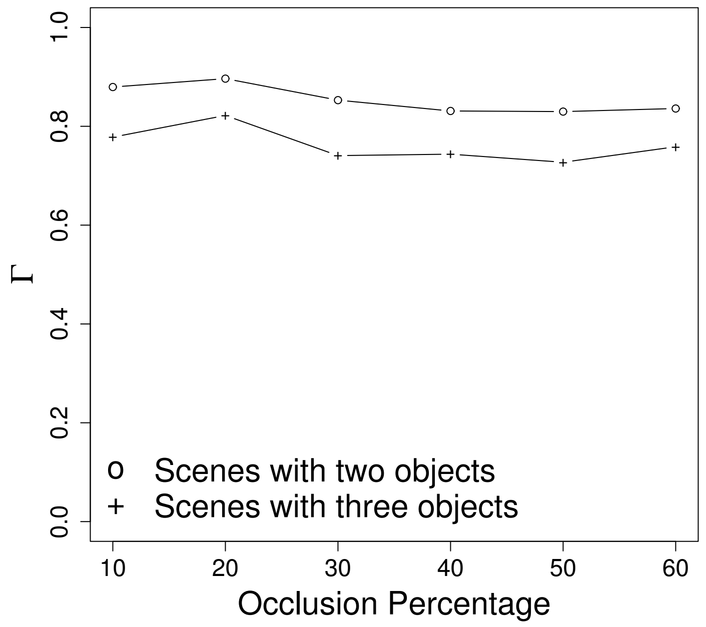


Figure 6.12: Results on localization of partially occluded objects. The average localization accuracy Γ is plotted for six discrete target occlusion percentages. The results show that for scenes with two and three objects Γ is always over 0.80 and 0.70, respectively. Additionally, in both cases Γ does not significantly decrease even at 60% occlusion.

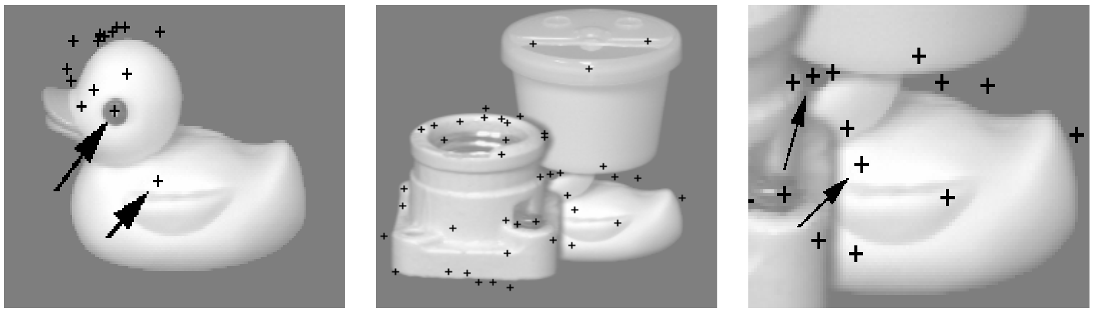
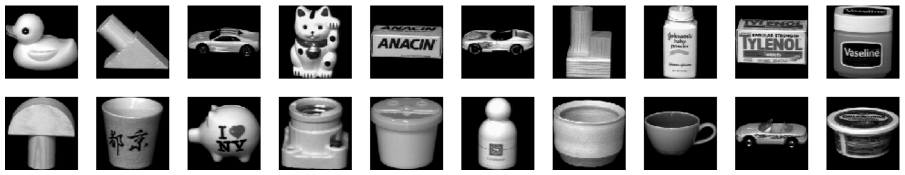


Figure 6.13: An example of a failed query. Left: Query object image, Middle: Target scene image, Right: Zoom in of the query object in the target scene. The SBCS features are depicted with crosses in all images. (All images are histogram scaled and cropped for clarity.) Since only two of the features in the query SBSC appear in the target SBCS, the query object cannot be accurately located. As this example illustrates, failures can be caused by lack of discriminating texture in the unoccluded regions of the target objects.


(a)

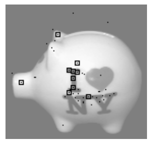

(b)

Figure 6.14: a) Frontal views of COIL-20 objects used in experiment. b) The features from object 13 are marked with black points, SBCS subset of size 10 used in experiment 1 marked with black squares.

algorithm was used to extract a subset of the image features with sizes ranging from 10 to 35 features. The inverse of the Euclidean distance between the feature locations was used as a similarity measure, and the significance field of the BLOB-RIDGE features was used as the stability.

Each of the original 20 objects were used as queries against the 20 transformed target objects. The HUT-EMD algorithm described in Section 6.3 was used to compute the pairwise distances between a query and a target. A query was considered a correct match if the target with the minimum distance was the query object under transformation. The matching task was conducted for a range of subset sizes, as well as the complete feature set. This was done to ensure that the feature subsets were representative, *i.e.*, they would have matching rates comparable to the matching rate for the full sets of query and target features.

Figure 6.15 shows the relative computational cost of finding the localization transformation using the HUT-EMD algorithm for various subset sizes. As the size of the feature subset decreases, the total computational time is greatly reduced with no appreciable loss of accuracy. In fact, in all cases the matching rates for the subsets were at least 98% of the matching rate for the full sets of query and target features. The subset selection method therefore produces a representative feature set for this task, while providing a significant decrease in the running time.

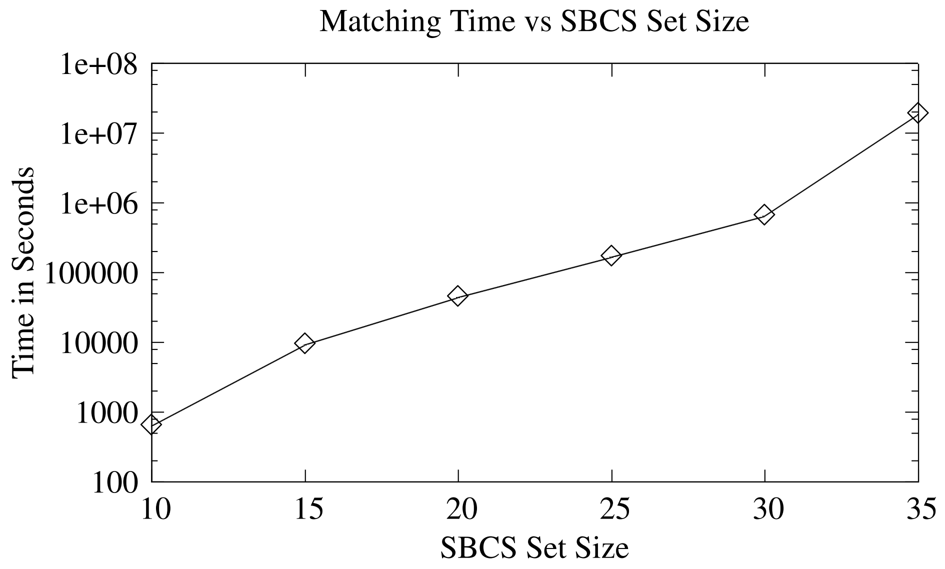


Figure 6.15: Matching time versus SBCS size. BLOB-RIDGE features were matched using HUT-EMD. See Section 6.2 for a description of BLOB-RIDGE and Section 6.3 for a description of HUT-EMD. Note that 10^7 seconds is 115 days.

6.12 SBCS Stability with Gaussian Noise

This experiment tested how the feature subsets created with the SBCS algorithm react with noise. The dataset of 20 query images from the previous experiment was used. The target images from the previous experiment were used to create 6 sets of images, with an increasing amount of Gaussian noise ($\sigma = 1, 2, 4, 8, 16, 32$) added to the pixel intensity values in each set.

Then features were extracted from the query and target images using the HARRIS, BLOB-RIDGE, and SIFT detectors. The SBCS algorithm was used to select subsets from the query and target features, reducing the size of the feature sets by half. The detector/similarity/stability combinations of HARRIS/XY/RES, BLOB-RIDGE/XY/SIG, SIFT/XY/SCALE, and SIFT/XY/PERTURB were used to produce 4 series of subsets. To match the HARRIS and BLOB-RIDGE features, the HUT-EMD algorithm was used, counting as a match the target with the lowest distance. To match the SIFT features the

MATCH algorithm was used, a match was defined as the target with the greatest number of matched features. HUT-EMD and MATCH are described in Section 6.3.

Figure 6.16 summarizes the results of the experiment. The figure shows the percent of queries that were correctly matched matched to the target image (the same image scaled and rotated with Gaussian noise added). The curves are relatively flat indicating that the SBCS of feature subsets are robust in the presence of Gaussian noise. Additionally, one can see that the combination of SIFT with MATCH significantly outperforms the other algorithms.

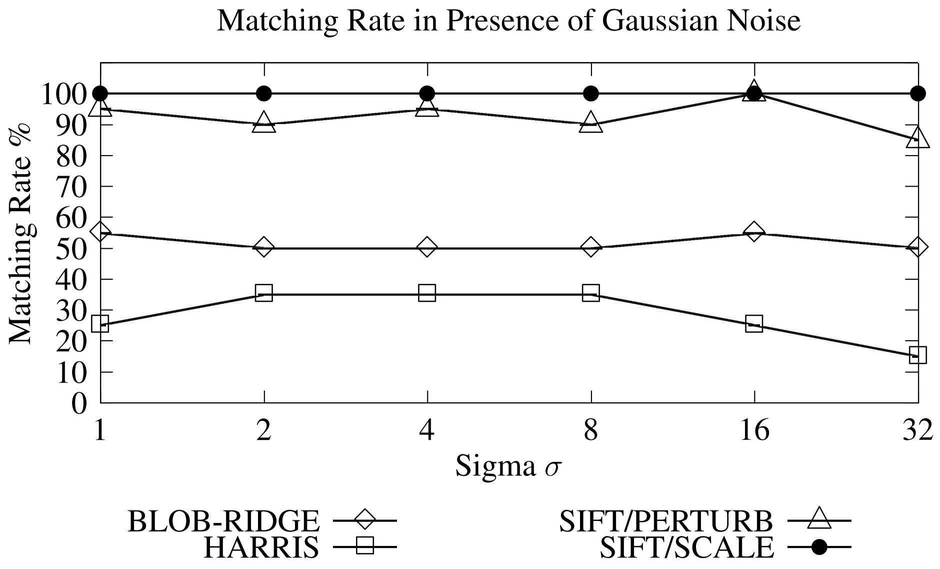


Figure 6.16: Matching performance with SBCS with HARRIS, BLOB-RIDGE, and SIFT feature detectors. The matching rate is the percentage of queries that were correctly matched to the target image (the same image scaled and rotated with Gaussian noise added). HARRIS and BLOB-RIDGE use the HUT-EMD matching algorithm, SIFT variants use MATCH algorithm supplied with SIFT. The SBCS is of size 50% of full set of features. SIFT/PERTURB uses perturb stability, SIFT/SCALE uses scale stability. See Section 6.5 for a description of stability measures.

6.13 Comparison with Other Subset Selection Techniques

In this experiment, the SBCS subset selection method is compared with other commonly used techniques of computing feature subsets. The techniques are compared by examining the performance in the context of object localization under significant occlusion. The SBCS method is compared to the subset selection techniques of thresholding the significance of the features and using K-means to create feature clusters. The results indicate the SBCS outperforms each of these techniques, and accurately localizes objects even in the presence of large amounts of occlusion.

6.13.1 Dataset

In order to test the performance of feature subset algorithms under occlusion, it was first necessary to extend the Drexel Object Occlusion Repository, DOOR [16], which allowed automation of the testing process. The new DOOR images (denoted series 6) are constructed by randomly scaling and rotating five input objects from the COIL-20 database [86] on a background image so that the degree of occlusion is exactly known. Ground-truth information sufficient to allow reconstruction permits accurate measurement of occlusion and localization accuracy by automated means. A breakdown of the occlusion rates and the number of examples is given in Table 6.1.

6.13.2 Localization Error

Provided with the ground-truth transformation, G, an error measure for a computed transformation, T, is defined. Let the error, e, be defined as

$$e = \frac{1}{ns} \sum_{i=1}^{n} \| Tp_i - Gp_i \|_2, \tag{6.1}$$

Occlusion Percent	**Object Examples**
$o = 0$	1063
$0.001 \leq o < 10$	228
$10 \leq o < 20$	123
$20 \leq o < 30$	105
$30 \leq o < 40$	89
$40 \leq o < 50$	100
$50 \leq o < 60$	85
$60 \leq o < 70$	95
$70 \leq o < 80$	89
$80 \leq o < 90$	93
$90 \leq o < 100$	330

Table 6.1: DOOR dataset series 6 breakdown of occlusion rates and number of object examples (e.g 89 objects are occluded between 30% and 40%). DOOR series 6 consists of 480 scenes each with 5 objects superimposed.

where n is the number of features in full set , s is the ground-truth scale factor, and p_i is a feature in the full set of model features. The notation, $\|\cdot\|_2$, denotes the Euclidean distance. Intuitively, the features are transformed by both T and G and then the average distance between where the transformation T places them versus where the ground-truth, G, places them is measured. The distance is then divided by the ground-truth scale to compensate for differences in scale (small distances in a small object are more significant than the same distance is a highly scaled object).

6.13.3 Algorithms

For the SBCS tests, the DSPACE similarity measure and I-PERTURB stability were used along with $\Lambda = \{\lambda_1 = 0.4625, \lambda_2 = 0.4625, \lambda_3 = 0.075\}$, which was determined empirically. For the threshold tests, a subset of SIFT features was selected based on their scales. The features were sorted by their scales and then either the top third of the features (threshold high) or the bottom third (threshold low) was selected.

For the K-Means tests an implementation by Mount [85] was used to cluster the features. From each cluster the feature was selected that was closest to the cluster centroid as the representative member, *i.e.*, the subset consists of the nearest neighbors to each of the cluster centroids. K-Means was tested in the descriptor space (K-Means DSPACE) and K-Means in x, y (K-Means XY). As a control, and for comparison purposes the tests were performed with the full set of SIFT features.

6.13.4 Methodology

SIFT features were extracted from the frontal view of each COIL-20 object, resulting in 20 sets of features. The algorithms described in Section 6.13.3 were then used to extract subsets of features that were a third of the original size.

Each of the DOOR series 6 occlusion scenes was processed and SIFT features were extracted. Then the ground-truth information for each scene was examined. For each object in the scene the MATCH algorithm supplied with SIFT was used to determine the corresponding matches between each of the feature subsets and the scene features. For example, if the scene contained object 7, we ran the MATCH program on the subsets of object 7 SIFT keys and the scene SIFT keys.

For each of the matchings, the calculated transformation (if the object was detected) was recorded as well as other statistics such as the error and the detection rate. The detection rate for a particular subset algorithm is defined as the number of detections divided by the number of detections with the full set of features. The rational behind using this measure is that it provides a frame of reference when considering the error measure, *i.e.*, using a subset algorithm with low error rates may not be desirable if the detection rate is too low.

6.13.5 Algorithmic Performance

Section 3.3.6 discussed the theoretical bounds of the SBCS algorithm, but how does it perform in practice? Of the 20 model objects, object 9 (Tylenol) had the most SIFT features with 212. Ten consecutive runs of the algorithm with $k_{min} = 12,\ k_{max} = 16$ averaged 44 seconds each on an Intel Core Duo @ 1.66GHz with 1 gig of ram. The average time to solve the SDP was 5.9 seconds with the balance of the time being spent on the rounding and parsing the results. Eight of the subsets had identical SBCS members and two of the subsets had slightly worse objective values (higher by 0.00015%). These two subsets had the same members and shared 14 of the 16 members of the other 8 subsets. The SBCS algorithm is typically run with 1500 roundings, and the experience is that the repeatability of the generated subsets can be improved by increasing the number of roundings.

It is important to note that as the size of the original feature set increases, the number of variables in the SDP increases quadratically. This does present an upper limit on the size of the feature sets that can be practically partitioned.

Although, the theoretical bounds for the SDP formulation have been given in Section 3.3.6, the bounds for the QP formulation must be tested empirically. A series of 20 tests were run to compare the performance of the SDP formulation with the quadratic programming (QP) approximation. For technical reasons both algorithms were run as minimizations. Figure 6.13.5 shows the results. The average ratio of objective values, QP/SDP was 1.003, and the ratio of execution times was 0.17. These results indicate that the QP approximation obtains objective value very close to those obtained with SDP, yet requires about one fifth of the execution time.

6.13.6 Results

Using the full set of features, 1321 detections of objects were obtained with an average error of 2.77 pixels. Figure 6.18 shows a graph of the detection rates from the algorithms

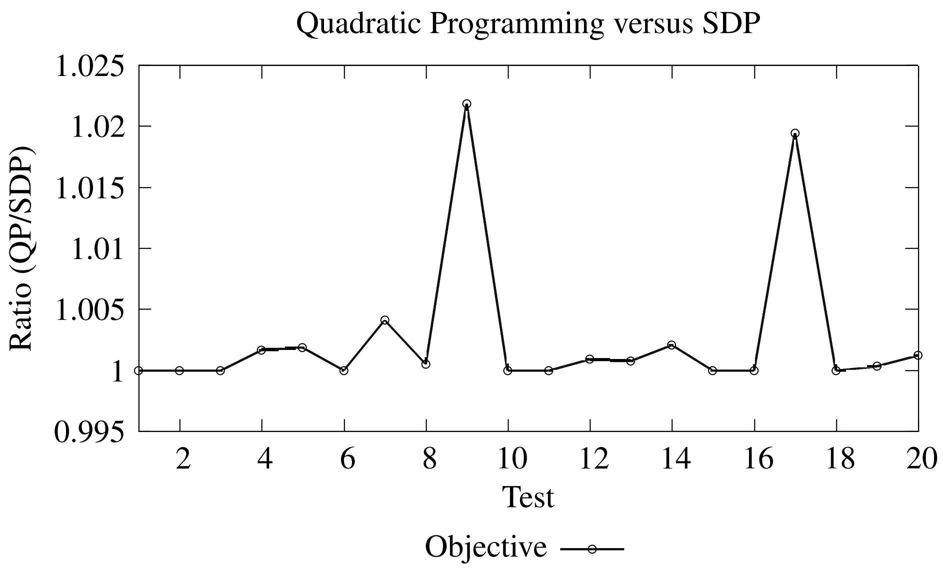


Figure 6.17: Comparative performance QP versus SDP approximation. The average ratio of objective values, QP/SDP was 1.003, and the ratio of execution times was 0.17. For technical reasons both algorithms were run as minimizations, thus lower values indicate better performance.

broken down by occlusion percentage. SBCS has comparable detection performance to the best of the other algorithms across the entire range of occlusion rates.

Figure 6.19 shows results sorted by number of detections. The first row shows the results for the full set of features. SBCS had a slightly higher detection rate in conjunction with the lowest average error rate. When error rates are normalized to eliminate the effects of increased detection rates, the SBCS algorithm has the lowest error rate.

Figure 6.20 shows examples of the localizations achieved with SBCS. The figure shows query objects with the full set of features and the SBCS. Note that a single point in the visualization can represent multiple SIFT features that are detected at the same location. The figure also shows the matchings, error rates, and percentage of occlusion.

It is perhaps surprising that the I-PERTURB stability measure contributes to the best results as intuitively one would think that stronger more stable features would be better. However, a little reflection reveals why this is not so. Stronger, more stable features, often have a larger support region, and a large support region is more likely to be obscured at higher occlusion rates. This conjecture is supported by the dismal detection performance shown by the Threshold High subset selection algorithm (see Figures 6.18 and 6.19).

The dataset allows the precise measurement of occlusion and the performance of feature subsets on the task of object localization. The results indicate that the SBCS provides improved performance over the other methods that were tested.

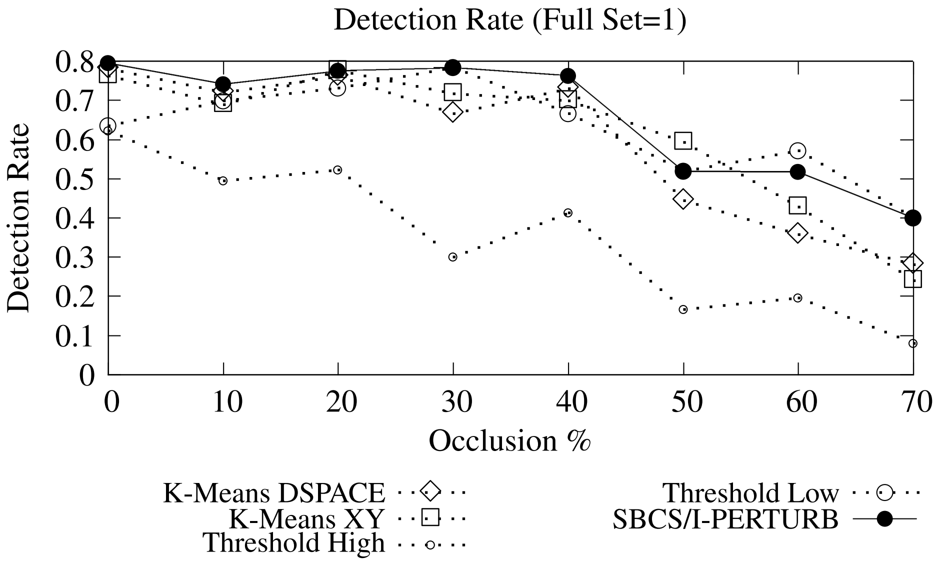


Figure 6.18: Comparison of detection rates of subset algorithms under varying amounts of occlusion. The detection rate is the number of detections divided by the number of detections obtained with the full set of features. SBCS/DSPACE/I-PERTURB has comparable performance over the entire range of occlusion. Data points include detections in a 10% range, *i.e.*, points at 60% includes detections under occlusion from 60-70%.

6.14 Object Localization in Real Images

Object localization under occlusion with synthetic images such as those used in the experiment described in Section 6.13 is considered to be an easier task than localization

Algorithm	**Detections**	**Rate**	**Avg. Error**	**Normalized Error**
Full	1321		2.7721	0.5701
SBCS	1104	83.5%	1.1312	0.5521
K-Means DSPACE	1084	82.0%	1.2844	0.6339
K-Means XY	1045	79.1%	2.0300	0.6252
Threshold Low	926	70.0%	2.8111	0.8912
Threshold High	889	67.2%	1.1655	1.1643

Figure 6.19: Results sorted by overall detection rate. The first row contains results from the full set of features. Rate is the percentage of full set detections. Average error is the average error in pixels adjusted to account for scaling. Normalized error is the average error over the detections common to all algorithms, *i.e.*, it shows the comparative error rates without taking increased detection rate into account. SBCS/DSPACE/I-PERTURB has a slightly higher detection rate than other subset selection algorithms, the lowest average error rates, and the lowest normalized error rate.

under occlusion with images that are real. This difficulty stems from factors including lighting, noise, reflections, and rotations in depth. However, because there is no source of ground-truth, the accuracy of localization and indeed the amount of occlusion, must be judged subjectively.

In order to test the suitability of the feature subsets selected by the SBCS algorithm, a dataset of real images of objects under occlusion was constructed. That is, objects were objects placed on a table and then photographed to form a set of query images. Then target scenes were created by arranging the query objects along with other objects on a table in various configurations and then subsequently photographing them.

The SIFT algorithm was used to extract sets of features from the query and target images. Using $\Lambda = \{\lambda_1 = 0.4625, \lambda_2 = 0.4625, \lambda_3 = 0.075\}$ with SBCS/DSPACE/I-PERTURB, subsets of the query features were computed that were a third of the size of the original query feature sets. The subsets of query features were used to localize the query objects in the target scenes and then the results were compared with localization using the full set of query features.

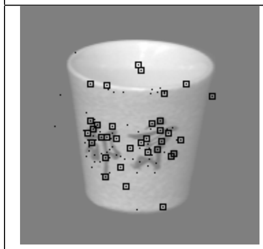
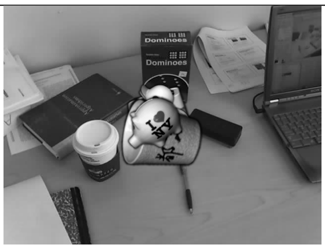
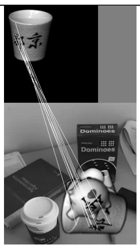
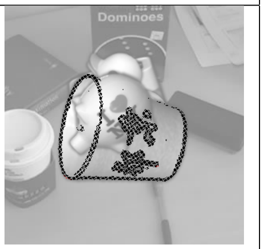
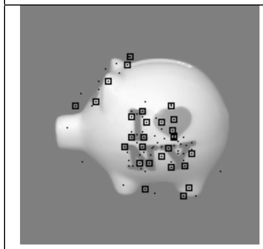
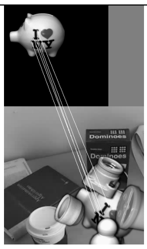
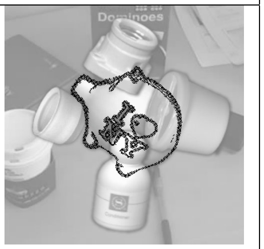
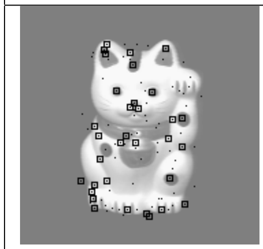
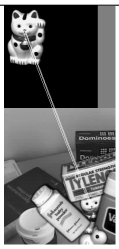
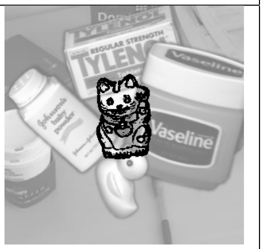
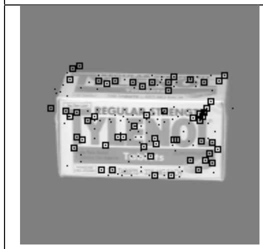
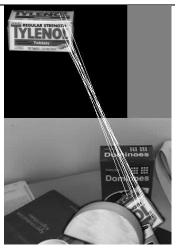
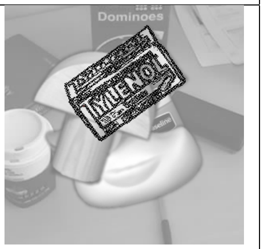
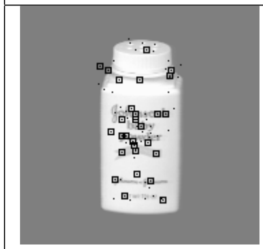
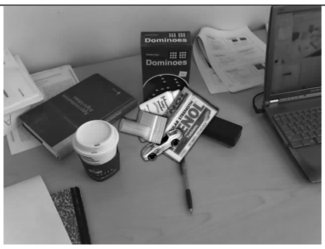
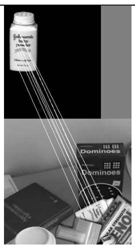
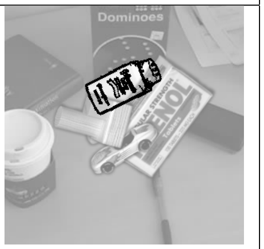
Features	Scene	Matching	Localization
			
			
			
			
			

Figure 6.20: Results from experiment with objects occluded by 50-75%. Column 1 shows the object with SIFT features as black points and the SBCS subset in black squares. Column 2 shows the scene. Column 3 shows the matching features. Column 4 shows a visualization of the localization accuracy.

Figure 6.21 shows some results of this experiment. Column one shows the query image, column two the scene, column three a closeup of the object of interest. Column four shows the localization found using the full set of query features, and the last column on the right shows the localization found using the subset of query features computed by our algorithm.

Examination of Figure 6.21 shows that the subset of features computed using our algorithm can effectively be used in a realistic setting for object localization under occlusion. In particular, the first three rows of the table show highly accurate localization under large amounts of occlusion. The results obtained with a subset of image features are comparable to those obtained with the full set of query features.

6.15 Conclusion

This chapter has presented an extensive set of experiments that explored the use of the SBCS algorithm in object recognition and localization tasks. Section 6.1 provided some background and an overview of the method where subsets of image features generated by the SBCS algorithm are used in object recognition and localization tasks. Section 6.2 detailed the feature detectors that were used in the experiments. Section 6.3 presented the methods that were used to match features in the query and target images and calculate the transformation between the matched features. Section 6.4 and 6.5 described the similarity and stability measures used in the experiments. Then Section 6.6 presented results that showed that it is possible to use a subset of image features to localize an object. Section 6.7 built on these results and showed localization of one object rotated in depth. Next Section 6.8 showed that multiple occluded objects could be localized using the method. Section 6.9 then explored pose estimation. In Section 6.10, the question of localization accuracy was addressed, a measure of accuracy was presented along with a synthetic database of occluded scenes accompanied by ground truth information. Section 6.11 then explored how the computational cost of image matching could be reduced through the use of subsets

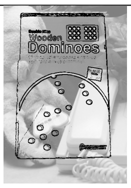
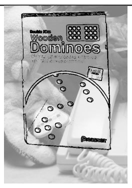
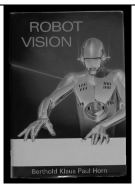
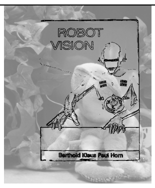
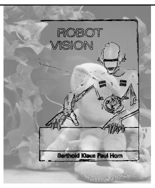
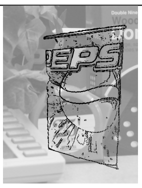
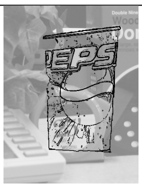
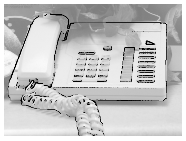
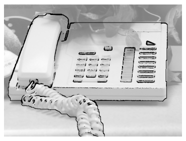
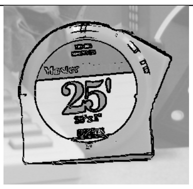
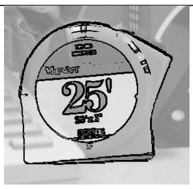
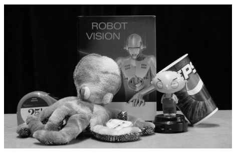
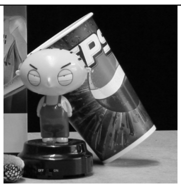
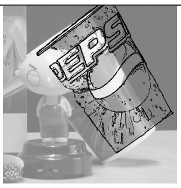
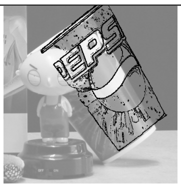
Query	Scene	Detail	Full Set	SBCS
				
				
				
				
				
				

Figure 6.21: Results comparing object localization with real images. Query column shows the query object. Scene, and Detail columns show the scene. The full set column shows the localization obtained with the full set of SIFT features, the SBCS column shows the localization obtained using a subset of the query features computed using our algorithm.

of image features. Next, in Section 6.12 the question of how the feature subsets created with the SBCS algorithm react with noise was addressed. Section 6.13 then presented a set of experiments that compared the SBCS with other subset selection techniques. Finally, Section 6.14 presented some results from localization experiments with real images.

7. Image Reconstruction

7.1 Introduction

Scale space interest points are features that encode the deep structure of the input images. Scale space interest points were first proposed by Iijima [42] and later described by Witkin [115], and many different types have been proposed since then [56, 66, 68, 82, 83, 94]. The increasing use of interest points in applications such as recognition, classification, and image editing has instigated research on interest points and their descriptive power. One way of evaluating the information content of interest points is through image reconstruction. Here, the motivation is that a sufficiently rich set of points should encode enough information about the structure of the original image to permit its accurate reconstruction.

To improve the accuracy of the reconstructions different types of interest points may be used, as they encode different aspects of the input image. Indeed, Lillholm and Nielsen [65, 87] showed that combining different types of interest points can improve the quality of the reconstruction. Typically different types of interest points are combined and then ordered by their strength or the differential TV-norm as described by Platel *et al.* [93], with a subset of the strongest points selected for reconstruction. Figure 7.1 shows an example reconstruction using this method and illustrates some of its drawbacks. Large regions of the reconstruction lack detail because none of the strongest points were located in those regions.

This chapter presents a method of selecting a subset from a combined set of scale space interest points for image reconstruction. This method is a generalization of preliminary studies of the canonical set framework [49] for image reconstruction. Section 7.2 presents an overview of the types of scale space interest points used in the experiments. Section 7.3 shows examples of reconstructions of a test image using different types of scale space

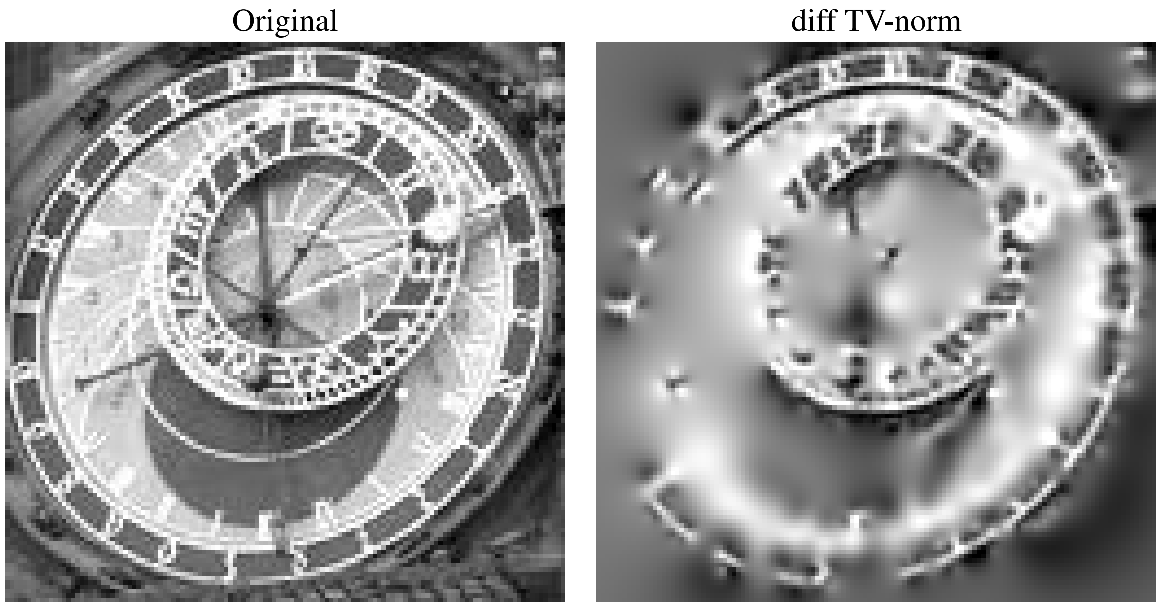


Figure 7.1: The image on the right is a reconstruction using the top 200 scale space interest points ordered by differential TV-norm [93].

points. Then Section 7.3.2 describes two measures of reconstruction quality that are used evaluate the reconstructions created during the experiments.

In the generalized framework which is presented in Section 7.4, spatial constraints are incorporated into the feature selection process. A proof of how the optimal spatial constraints can be computed in polynomial time is presented in Section 7.4.1. In Section 7.5, results from a comprehensive set of experiments on a suite of test images are shown.

7.2 Scale Space Interest Points

Scale space theory is the theory of apertures, through which humans and machines observe the world. For computer vision systems, the notion of aperture can be introduced as *blurring* the high resolution image with a kernel of a certain width [42, 55, 115]. The linear scale space representation $u : \mathbb{R}^d \times \mathbb{R}_+ \to \mathbb{R}$ of a continuous image $f : \mathbb{R}^d \to \mathbb{R}$ is defined as the solution of the heat equation:

$$\begin{cases} \frac{\partial}{\partial s} u = \Delta u \\ \lim_{s \downarrow 0} u(\cdot, s) = f(\cdot) \end{cases} \tag{7.1}$$

where s denotes the scale. The unique solution to this equation leads to convolution with a Gaussian kernel. Spatial derivatives of the image can be calculated by convolution with a derivative of a Gaussian:

$$\partial_{\nu^1,\ldots,\nu^n} u(x, y, s) = (\partial_{\nu^1,\ldots,\nu^n} G_s * f)(x, y) \tag{7.2}$$

where f is the original image, G_s a Gaussian of scale s, and $\nu^1, \ldots, \nu^n$ the spatial indices (for a 2D image this can be any combination of x and y). For the remainder of this chapter, the following the short notation $u_{s,\nu^1 \ldots \nu^n}$ is used for $\partial_{\nu^1,\ldots,\nu^n} u(x, y, s)$.

A scale space of a 2D image is in fact a 3D volume with the scale s as the third dimension. In the scale space representation of an image, several special types of interest points can be identified. These points contain important structural information of the image and can be used for image matching [83, 94] and reconstruction [47, 65]. In the remainder of this section, 10 types of commonly used scale space interest points are presented.

7.2.1 Laplacian Blobs

Scale space blobs [66] are defined as the positive local maxima (or negative local minima) in space and scale of the normalized Laplacian of the image:

$$\operatorname*{lmax}_{x,y,s}\{|s^\gamma (u_{s,xx} + u_{s,yy})|\} \tag{7.3}$$

with lmax the local maximum and using γ-normalization with $\gamma = 1$. The γ-normalization with scale invariance is discussed in detail by Florack and Kuijper [25]. Blobs can be ordered in strength by the magnitude of the response of their respective filters [87, 65]. Figure 7.2 shows a synthetic image and the detected Laplacian blobs depicted as points. The circles surrounding the points indicate the scale of the interest points.

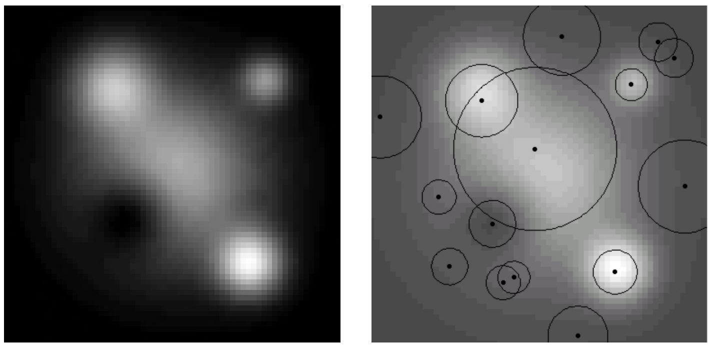


Figure 7.2: Laplacian blobs

7.2.2 Hessian Blobs

Alternatively scale space blobs can be defined as the local maxima of the squared normalized determinant of the Hessian.

$$\underset{x,y,s}{\text{lmax}}\{s^{4\gamma}(u_{s,xx}u_{s,yy} - u_{s,xy}^2)^2\} \tag{7.4}$$

Again using γ-normalization with $\gamma = 1$. Figure 7.3 shows a synthetic image and the detected Hessian blobs depicted as points. The circles surrounding the points indicate the scale of the interest points.

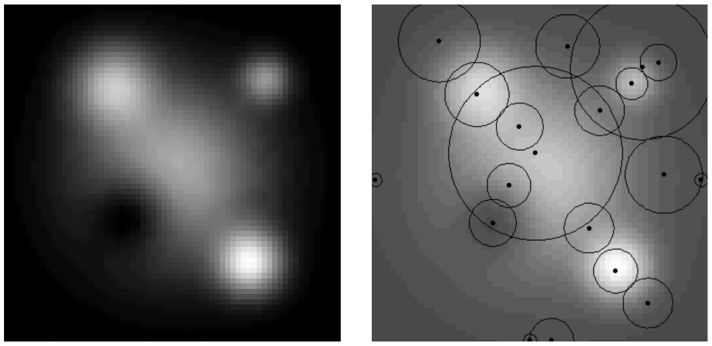


Figure 7.3: Hessian blobs

7.2.3 Corner Points

Corner points in the image are defined as points with high curvature and high intensity gradient:

$$\operatorname*{lmax}_{x,y,s}\{|s^{2\gamma}(2u_{s,x}u_{s,xy}u_{s,y} - u_{s,xx}u_{s,y}^2 - u_{s,x}^2u_{s,yy})|\} \tag{7.5}$$

using γ-normalization with $\gamma = 7/8$, following Lindeberg *et al.* [67]. Note that for ordering the corner points in strength, the magnitude of the corresponding filter response has to be normalized with $\gamma = 1$ to make magnitude values at different scales comparable. Figure 7.4 shows a synthetic image and the detected corner points. The circles surrounding the points indicate the support regions.

7.2.4 Edge Points

Edge points are defined by the following two constraints [65, 87]:

$$\begin{cases} s^{\gamma}u_{ww} = 0 \\ \operatorname*{lmax}_{s}\{s^{\gamma/2}u_w\} \end{cases} \tag{7.6}$$

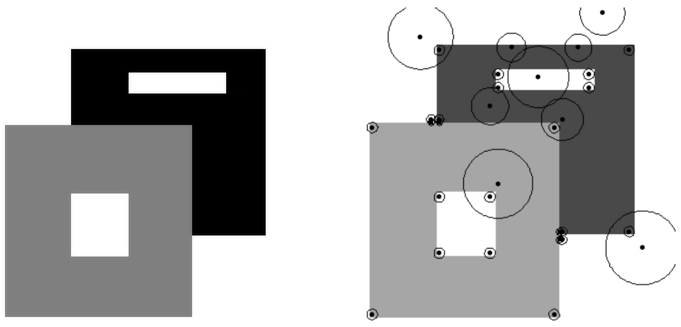


Figure 7.4: Corner Points

using γ-normalization with $\gamma = 1/2$ [67]. Here u_w is the first order derivative in the gradient direction and u_{ww} the second order derivative in the gradient direction. For edge strength, the gradient magnitude is used re-normalized with $\gamma = 1$. Figure 7.5 shows an example image and the detected edge points. The circles surrounding the points indicate the support regions.

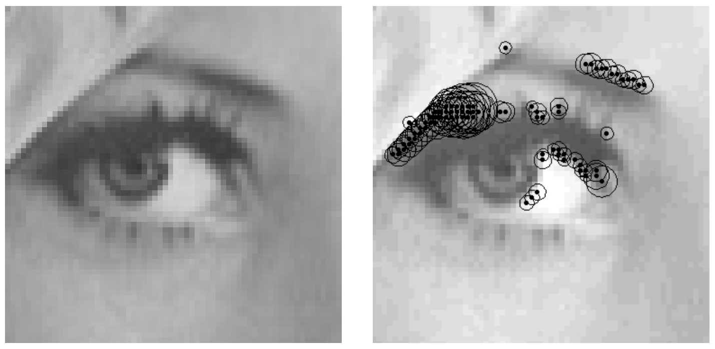


Figure 7.5: Edge Points

7.2.5 Ridge Points

Ridge points are defined as the local extrema of the *square of the γ-normalized principal curvature difference* [35, 67]:

$$\operatorname*{lmax}_{x,y,s}\{s^{2\gamma}((u_{s,xx} - u_{s,yy})^2 + 4u_{s,xy}^2)\} \tag{7.7}$$

Note that for ordering the ridge points in strength, again the magnitude of the corresponding filter response has to be normalized with $\gamma = 1$ to make magnitude values at different scales comparable. Figure 7.6 shows an example image and the detected ridge points. The circles surrounding the points indicate the support regions.

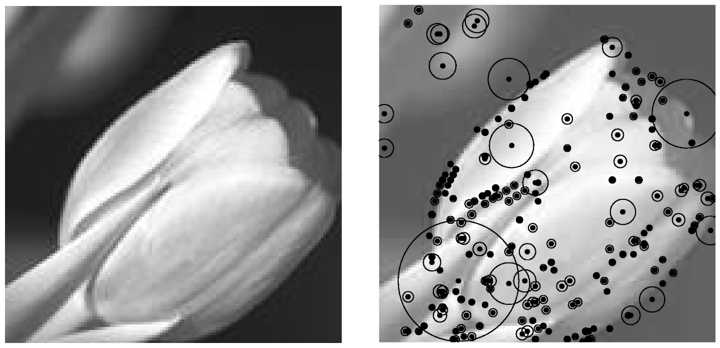


Figure 7.6: Ridge Points

7.2.6 Top Points

Top points are defined by:

$$\begin{cases} \nabla u_s = (u_{s,x}, u_{s,y})^T = \mathbf{0} \\ \det \mathcal{H}(u_s) = u_{s,xx}u_{s,yy} - u_{s,xy}^2 = 0 \end{cases} \tag{7.8}$$

where $\mathcal{H}(u_s)$ is the 2^{nd} order Hessian matrix defined by:

$$\mathcal{H}(u_s) = \begin{pmatrix} u_{s,xx} & u_{s,xy} \\ u_{s,xy} & u_{s,yy} \end{pmatrix} \tag{7.9}$$

Platel and Kanters showed that top points can be rank-ordered by a stability norm called differential (quadratic) TV-norm [94, 47]. Figure 7.7 shows a synthetic image and the detected top points. The circles surrounding the points represent the scale.

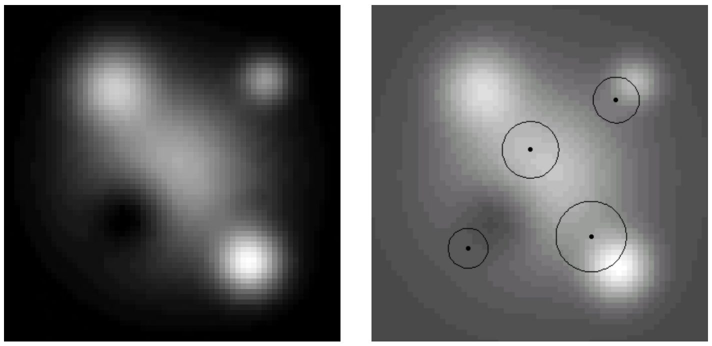


Figure 7.7: Top Points

7.2.7 Top Points of the Laplacian

It has been shown that top points of the Laplacian of an image (versus top points of the gray level image itself) can be used for image matching [94] and reconstruction [46]. These points are defined by:

$$\begin{cases} u_{s,xxx} + u_{s,yyx} = 0 \\ u_{s,xxy} + u_{s,yyy} = 0 \\ (u_{s,xxxx} + u_{s,yyxx})(u_{s,xxyy} + u_{s,yyyy}) - (u_{s,xxxy} + u_{yyxy})^2 = 0 \end{cases} \tag{7.10}$$

Laplacian top points can be seen as points in scale space where one Laplacian extremum blob and one Laplacian saddle merge into one blob. Instead of describing the behavior of local extrema in scale space, Laplacian top points describe the behavior of blobs through scale. Figure 7.8 shows a synthetic image and the detected Laplacian top points. The circles surrounding the points represent the scale.

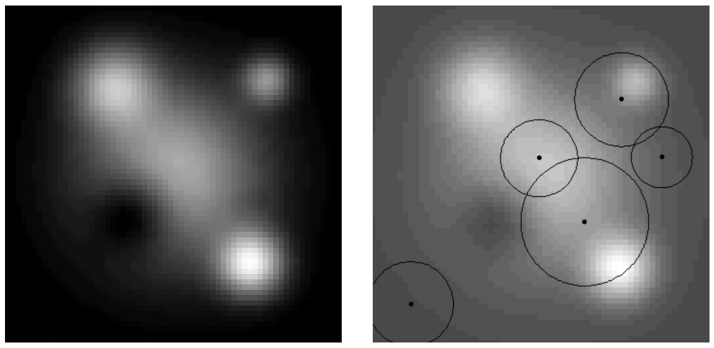


Figure 7.8: Laplacian Top Points

7.2.8 Scale Space Saddle Points

Koenderink [56] and Kuijper et al. [60, 61, 62] introduced scale space saddle points, which are defined by:

$$\begin{cases} u_{s,x} = 0 \\ u_{s,y} = 0 \\ u_{s,xx} + u_{s,yy} = 0 \end{cases} \tag{7.11}$$

Figure 7.9 shows a synthetic image and the detected Scale space saddle points. The circles surrounding the points represent the scale.

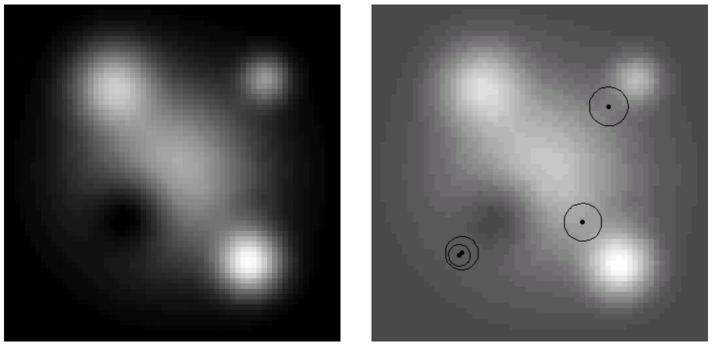


Figure 7.9: Scale Space Saddle Points

7.2.9 Hessian-Laplace Points

Mikolajczyk and Schmid [84, 83] introduced a hybrid method where the local spatial maxima of the square of the determinant of the Hessian matrix (introduced by Lowe [73] to eliminate edge response) are combined with the local scale maxima of the Laplacian. For strength the Laplacian is used.

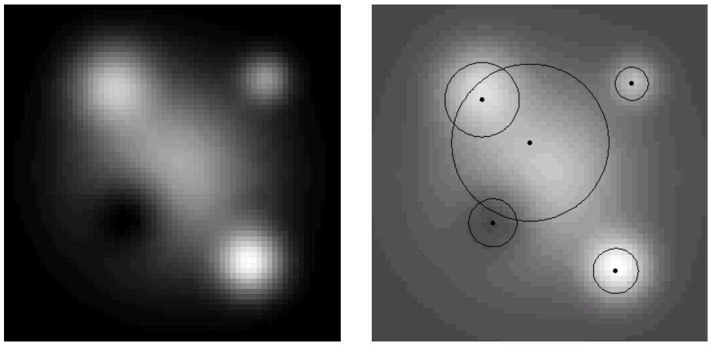
$$\begin{cases} \operatorname*{lmax}_{x,y}\{s^{4\gamma}(u_{s,xx}u_{s,yy} - u_{s,xy}^2)^2\} \\ \operatorname*{lmax}_{s}\{s^{2\gamma}(u_{s,xx} + u_{s,yy})\} \end{cases} \tag{7.12}$$

Figure 7.10 shows a synthetic image and the detected Hessian-Laplace points. The circles surrounding the points represent the scale.

7.2.10 Harris-Laplace Points

Mikolajczyk and Schmid [82] also introduced a scale adapted version of the Harris corner detector [37]. Consider the scale-adapted second moment matrix:

$$\mu(s_D, s_I) = s_D G_{s_I} * \begin{pmatrix} u_{s_D,xx} & u_{s_D,xy} \\ u_{s_D,xy} & u_{s_D,yy} \end{pmatrix} \tag{7.13}$$



Synthetic Image Hessian-Laplace Points

Figure 7.10: Hessian-Laplace Points

with s_D the *differentiation* scale, s_I the *integration* scale, and G_{s_I} a Gaussian at scale s_I. The matrix describes the gradient variation in a local neighborhood of a point. The Harris measure [37] combines the trace and determinant of this matrix as a measure for cornerness. Scale selection is based on the local maxima over scale of the Laplacian. The scale adapted Harris-Laplace points are defined as:

$$\begin{cases} \underset{x,y}{\text{lmax}}\{\det(\mu(s_D, s_I)) - \alpha \text{trace}^2(\mu(s_D, s_I))\} \\ \underset{s}{\text{lmax}}\{s^{2\gamma}(u_{s,xx} + u_{s,yy})\} \end{cases} \tag{7.14}$$

Figure 7.11 shows an example image and the detected Harris-Laplace points. The circles surrounding the points represent the scale.

7.3 Reconstruction and Selection Based on Differential TV-norm

Using scale space interest points and local derivatives up to 4^{th} order in these points, an approximation can be made of the original image from which the points were extracted. The reconstruction algorithm used for this purpose is based on the Sobolev reconstruction algorithm proposed by Janssen *et al.* [44]. The reconstruction algorithm finds an image that is as smooth as possible, with exactly the same derivatives at the scale space interest points

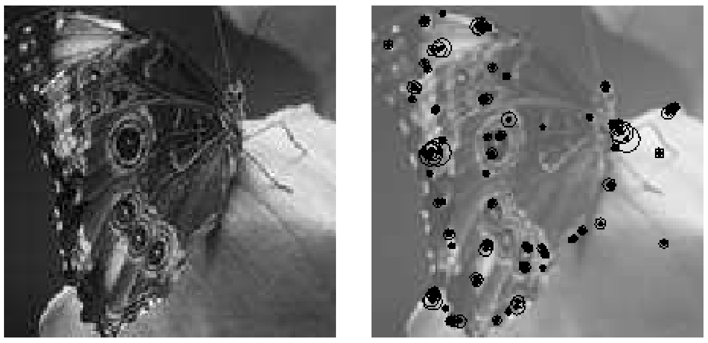


Figure 7.11: Harris-Laplace Points

using an orthogonal projection in a Sobolev space. Figure 7.12 shows the reconstruction results for a test image (Figure 7.1) for different types of scale space interest points.

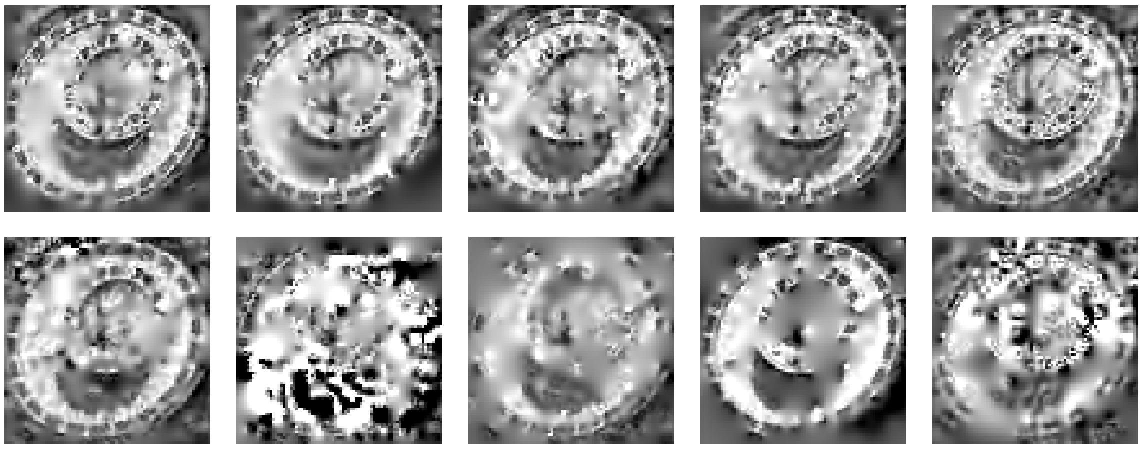


Figure 7.12: Reconstructions of a test image (Figure 7.1) using the 200 strongest top points of the Laplacian, Hessian blobs, corner points, Laplacian blobs, top points, ridge points, Hessian Laplace points, Harris Laplace points, edge points and scale space saddles, left to right and top to bottom respectively.

As can be seen from the reconstruction results in Figure 7.12, different types of interest points capture different aspects of image structure. Therefore, combining different types of interest points should naturally improve the reconstruction quality. The question is how to combine the different types of points. One possible option is to combine all types of points in a large set and re-order all points using their differential TV-norm [47, 94, 93] or a similarly defined strength measure.

7.3.1 Differential TV-norm

The differential TV-norm measures the local structure in a small neighborhood around the interest point. The amount of structure contained in a *spatial* area around a critical point can be quantified by the *total* (*quadratic*) *variation* (TV) norm over that area [9]. By using a spatial Taylor series around a considered critical point the TV-norm simplifies to Eqn. (7.15) which is referred to as the *differential TV-norm* [94],

$$\mathrm{diff\ tv} = 4s^2(u_{s,xx}^2 + u_{s,yy}^2 + 2u_{s,xy}^2). \tag{7.15}$$

The result of the 200 strongest combined interest points ordered by differential TV-norm of a test image and the corresponding reconstruction is shown in Figure 7.13.

Note that the quality of the combined scale space interest point reconstruction is lower than some of the separate interest point reconstructions shown in Figure 7.12. The reason for this is that in the combined point set many points that are close to each other will share a similar differential TV-norm (or other strength measure). Using strength measures as the sole criteria for ordering interest points may result in selecting points close to each other and consequently poor reconstruction results since these points will contain much redundant information.

7.3.2 Reconstruction Quality

In order to compare the results of the experiments, objective measures are necessary to evaluate the quality of the reconstruction. Ideally this measure should reflect the human

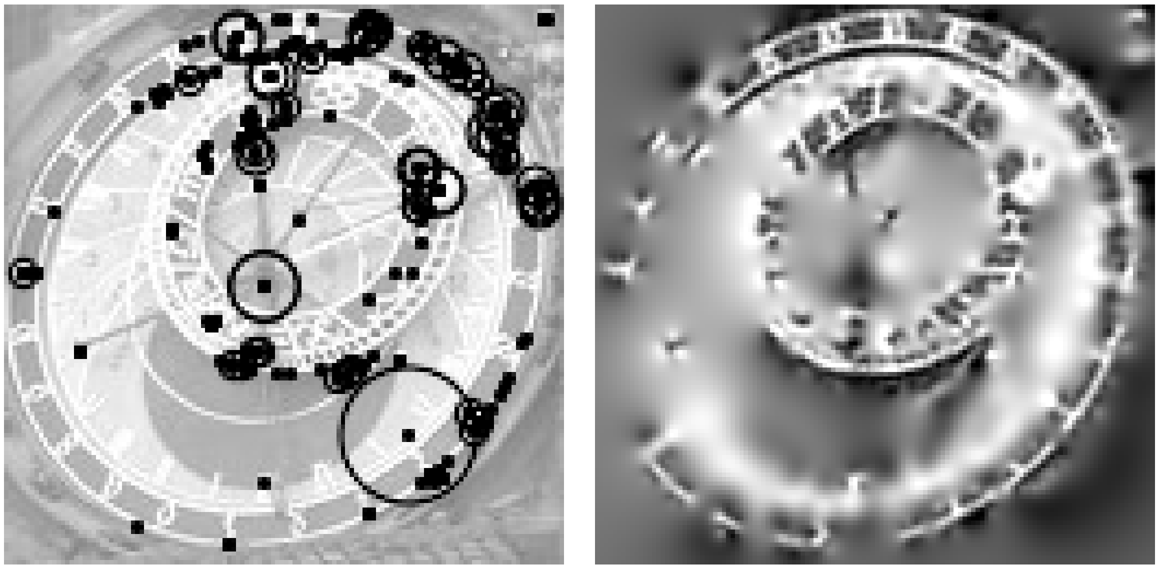


Figure 7.13: Left: 200 strongest combined scale space interest points of a test image ordered by differential TV-norm. Right: Reconstruction of a test image using the 200 strongest combined scale space interest points ordered by differential TV-norm.

observer's notion of quality. The Structural Similarity Measure (SSIM) of Wang *et al.* [113] is one metric that can be used to determine the quality of reconstructions. The SSIM incorporates a human visual system model of degradation of structural information.

Another possible metric is the Multi Scale Differential Error (MSDE) which considers the differential structure of the difference between two images at several scales simultaneously, and like the human visual system is sensitive to the differential structure of an image at multiple scales. A detailed motivation and evaluation of this error measure is presented by Kanters *et al.* [48]. For R different scales, let $\Gamma = \{\sigma_1, \ldots, \sigma_R\}$, then the gradient magnitude error map $\Psi_{f,g}$ between image f and reconstructed image g is defined as:

$$\Psi_{f,g}[i,j] = \sqrt{\frac{1}{R}\sum_{\sigma\in\Gamma}\left(|\sigma\nabla f_\sigma\left[i,j\right]| - |\sigma\nabla g_\sigma\left[i,j\right]|\right)^2} \tag{7.16}$$

with $|\nabla_{f,\sigma}[i,j]|$ at point $[i,j]$ the gradient magnitude of image f at scale σ. The Multi Scale

Differential Error is defined as:

$$MSDE(f,g) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \Psi_{f,g}[i,j]. \tag{7.17}$$

7.4 Using the SBCS for Reconstruction

The motivation for using canonical sets to select subsets of scale space points stems from observation of the locations of points selected through ordering of the differential TV-norm. Visual inspection of the reconstruction on the right of Figure 7.13 reveals large areas that lack detail. Looking at the locations of the scale space points in the left image of Figure 7.13 reveals the cause; numerous points are clumped together. In order to take into account the spatial distance between points as well as the strength of the points, we select canonical subsets of the combined point sets. Recall from Section 2.7.3 that the SBCS is a subset of points with special properties namely:

1. Data points in the canonical set are minimally similar,
2. Data points in the canonical set are maximally similar to data points not in the canonical set,
3. Data points in the canonical set are maximally stable,
4. The size of the canonical set is as least k_{min} and at most k_{max}.

In the context of image reconstruction, the data points are the scale space interest points and the stability is the differential TV-norm. The inverse Euclidean distance between interest point locations is used as a similarity measure. Recall also that the SBCS can be approximated by the semidefinite program,

$$\begin{array}{rl} \textbf{(SBCS):} & \\ \text{Maximize} & \mathcal{C} \bullet \mathcal{X} \\ \text{Subject to} & \mathcal{D}_i \bullet \mathcal{X} \geq 0, \ \forall\, i = 1 \ldots, m, \\ & \mathcal{X} \succeq 0, \end{array}$$

where

$$\mathcal{C} \;=\; \lambda_1 \begin{bmatrix} \tilde{\mathbf{0}} & -\frac{1}{4}\mathbf{d} \\ -\frac{1}{4}\mathbf{d}^T & \frac{1}{2}w_\Sigma \end{bmatrix} + \lambda_2 \begin{bmatrix} -\frac{1}{4}\mathcal{W} & \hat{\mathbf{0}} \\ \hat{\mathbf{0}}^T & \frac{1}{4}w_\Sigma \end{bmatrix} + \lambda_3 \begin{bmatrix} \tilde{\mathbf{0}} & \frac{1}{4}\mathbf{t} \\ \frac{1}{4}\mathbf{t}^T & \frac{1}{2}t_\Sigma \end{bmatrix}, \quad (7.18)$$

and $\tilde{\mathbf{0}}$ is an $n \times n$ matrix of zeros, $\hat{\mathbf{0}}$ is an all zeros column vector in $\mathbb{R}^n$, $\mathbf{d}$ is a column vector in $\mathbb{R}^n$ whose i^{th} entry has value $d_i = \sum_{j=1}^{n} \mathcal{W}_{ij}$, $\mathcal{W}$ is the $n \times n$ similarity matrix, $w_\Sigma = \sum_{i,j} \mathcal{W}_{ij}$, $\mathbf{t}$ is a column vector in $\mathbb{R}^n$ whose i^{th} entry is t_i, and $t_\Sigma = \sum_{i=1}^{n} t_i$. Constraint matrices $\mathcal{D}_1, \ldots, \mathcal{D}_{n+1}$ are as described in Eqn. (3.22), $\mathcal{D}_{n+2}$ is as described in Eqn. (3.23), $\mathcal{D}_{n+3}$ is as described in Eqn. (3.24), and $m = n + 3$. The coefficients, $\lambda_1 \geq 0, \lambda_2 \geq 0$, and $\lambda_3 \geq 0$ are parameters such that $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

7.4.1 Incorporating Distance Lower Bound Constraints

In this section it is shown that the canonical set framework is flexible enough for incorporating distance lower bound constraints among the members of the canonical set. Specifically, in addition to the four properties, it is desired that for a given radius parameter r, if $p_i \in \mathcal{P}^*$ then $p_j \notin \mathcal{P}^*$ if the distance between p_i and p_j is less than or equal to r. The approach is similar to the adaptive non-maximal suppression of Brown *et al.* [8]. This set of constraints will prevent the selection of canonical features that are spatially too close to each other. The structure of *forbidden* pairs with respect to $\mathcal{P}$ will be encoded in the form of a $\{0,1\}$ binary matrix $\mathcal{F} = \mathcal{F}(r)$ where entry $\mathcal{F}_{ij} \in \{0,1\}$. Let $\mathcal{F}_{ij} = 1$ if the distance between features p_i and p_j is less than or equal to r and zero otherwise. Intuitively, the matrix $\mathcal{F}$ encodes the forbidden pairs, that is, if two features are closer than r to each other then at least one of them should be outside the canonical set. This constraint can be stated as:

$$\frac{1}{4}\sum_{i,j}\mathcal{F}_{ij}(1+y_iy_{n+1})(1+y_jy_{n+1}) \;=\; 0. \tag{7.19}$$

Using the fact that $\mathcal{F}$ is symmetric and $y_i^2 = 1$, for each $1 \leq i \leq n+1$, the constraint (7.19) can be restated as:

$$\frac{1}{4}\sum_{i,j}\mathcal{F}_{ij} + \frac{1}{4}\sum_{i,j}\mathcal{F}_{ij}y_iy_j + \frac{1}{2}\sum_{i=1}^{n}y_iy_{n+1}\sum_{j=1}^{n}\mathcal{F}_{ij} \;=\; 0. \tag{7.20}$$

Letting $f_\Sigma = \sum_{i,j}\mathcal{F}_{ij}$, and defining $\mathbf{f}$ as a column vector in $\mathbb{R}^n$ whose i^{th} entry is equal to $f_i = \sum_{j=1}^{n}\mathcal{F}_{ij}$, this latter constraint may be expressed as

$$\hat{\mathcal{F}} \bullet \mathcal{X} = 0, \text{ where } \hat{\mathcal{F}} = \begin{bmatrix} \frac{1}{4}\mathcal{F} & \frac{1}{4}\mathbf{f} \\ \frac{1}{4}\mathbf{f}^T & \frac{1}{4}f_\Sigma \end{bmatrix}. \tag{7.21}$$

The new SDP formulation for the SBCS with forbidden relations can be stated as

SBCS2:

$$\text{Minimize} \quad \mathcal{C} \bullet \mathcal{X} \tag{7.22}$$

$$\text{Subject to} \quad \mathcal{D}_i \bullet \mathcal{X} \geq 0,\ \forall\, i = 1\ldots, m, \tag{7.23}$$

$$\mathcal{X} \succeq 0, \tag{7.24}$$

where $\mathcal{C}$ is as described in (7.18), $m = n + 4$, and the first $n + 3$ constraint matrices are as described before in Eqn. (3.22), and constraint matrix $\mathcal{D}_{n+4}$ is (7.21). The approximate solution for this modified formulation can be estimated in a similar manner to the original SBCS problem.

Computing the Optimal Radius

It is desirable to have the value of r to be as large as possible, thereby spreading the SBCS2 members as far apart as possible. Without loss of generality, assume that the distances between all p_i and p_j are normalized to fall between zero and one. To find the optimal value of r, note that when $r = 0$ the formulation reduces to the standard SBCS formulation, *i.e.* the matrix $\mathcal{F}$ is all zeros. On the other hand if $r = 1$, the matrix $\mathcal{F}$ is all ones and the problem is infeasible for most purposes since only one point can be in the canonical set. Next it is shown how the optimal r can be found in polynomial time.

Lemma 7.4.1. *Let r be some value such that the SBCS2 algorithm is feasible. Then any value $r - \delta$ where $0 < \delta < r$ is feasible.*

Proof. Any pair that is forbidden by r is also forbidden by $r - \delta$. This is clearly the case since, if the distance between two set members, p_i and p_j is less than $r - \delta$ it is also less than r. Thus an instance using a minimum radius of $r - \delta$ is less constrained than an instance using r. $\square$

Lemma 7.4.2. *There are at most $n^2/2$ values for r that give distinct matrices $\mathcal{F}$, where $n = |\mathcal{P}|$.*

Proof. Since matrix $\mathcal{F}$ is symmetric and $\mathcal{F}_{ij} = 1$ if the distance between features p_i and p_j is less than or equal to r and zero otherwise, the number of distinct $\mathcal{F}$ matrices is bounded by the number of distinct distances, which can be no more that $n^2/2$. $\square$

Lemma 7.4.3. *The number of ones in the matrix $\mathcal{F}$ increases monotonically as the value of r increases.*

Proof. Let d_{ij} be the distance between p_i and p_j. Clearly, if $d_{ij} \leq r$ then $d_{ij} \leq r + \delta$, where $\delta > 0$. □

Theorem 7.4.4. *The optimal value for r can be found by running the SBCS2 algorithm at most $O(\log n)$ times.*

Proof. Let r^* be the optimal value for r. By lemma 7.4.2 there are at most $n^2/2$ possible values for r that must be searched, by lemma 7.4.1 and lemma 7.4.3 we know for any particular r whether $r^* < r$ or $r^* > r$. We can thus perform a binary search to find r^*, running the SBCS algorithm at most $O(\log n)$ times. □

The algorithm for computing the approximate SBCS with forbidden pairs, which is denoted SBCS3 thus consists of running the SBCS2 algorithm at most $O(\log n)$ times, performing a binary search to find the optimal radius r. Since SDP algorithms such as SBCS2 run in polynomial time, it follows that SBCS3 also runs in polynomial time.

7.5 Experiments

For these experiments, the 12 test images shown in Figure 7.14 were used. For each image, the 10 types of scale space interest points described in Section 7.2 were extracted and combined into one set ordered by differential TV-norm. The total number of interest points extracted ranged from 7160 for the tulip image (image number 12 in Figure 7.14) to over 10438 for the clock image (image number 9 in Figure 7.14).

From each of the combined point sets, the top 200 points ordered by differential TV-norm, denoted DTVNORM, were extracted. Then reconstructions were created, and the MSDE and SSIM were measured. Next the size of the combined feature sets was reduced

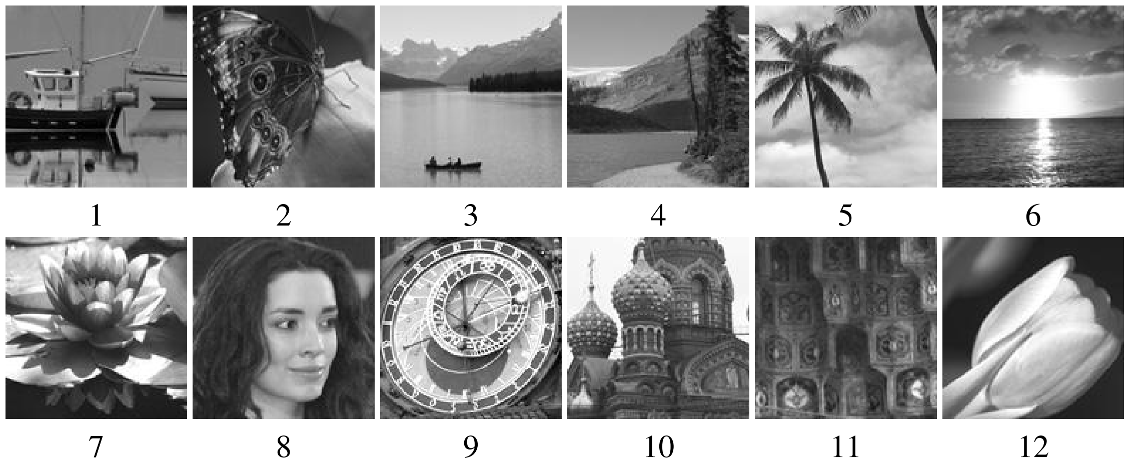


Figure 7.14: Images used in experiments, all images are 128 by 128 pixels.

by discarding points within a radius of 3 pixels of one with a higher differential TV-norm. The size of the resulting filtered combined sets was in the order of 1000 interest points.

Recall that the SBCS algorithm requires weighting parameters to control the relative importance of the multiple objectives. The Pareto weightings $\Lambda = \{\lambda_1, \lambda_2, \lambda_3\}$ were parametrized as

$$\lambda_1 = \frac{\alpha}{2} \tag{7.25}$$

$$\lambda_2 = \frac{\alpha}{2} \tag{7.26}$$

$$\lambda_3 = 1 - \alpha. \tag{7.27}$$

The SBCS3 algorithm was run on each of the reduced combined interest point sets, varying the convexity parameter, α, between 0 and 1 in 0.1 increments and subsets of 200 points were selected. Reconstructions were created from each of the subsets and the MSDE and SSIM were measured. The graph in Figure 7.15 shows the ratio of SSIM measurements for SBCS3/DTVNORM, values above 1 indicate the SBCS3 algorithm produces better reconstructions. Figure 7.16 shows a similar graph for the MSDE rates, in this case

DTVNORM/SBCS3. Again, values above 1 indicate the SBCS3 algorithm gives better reconstructions.

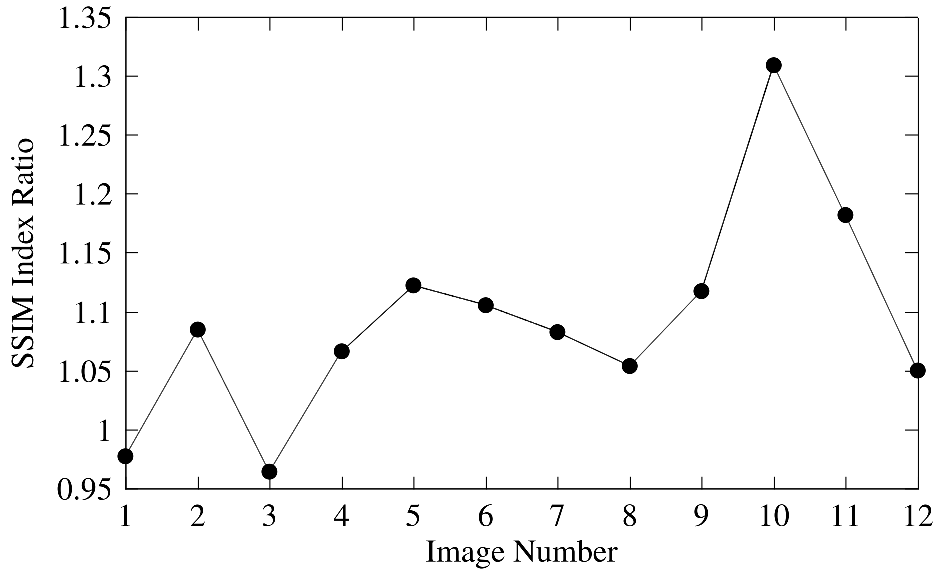


Figure 7.15: Ratio of SSIM measurements for SBCS3/DTVNORM (values above 1 indicate the SBCS3 algorithm produces better reconstructions).

Examination of the graphs in Figures 7.15 and 7.16 show that subsets selected based on the canonical set algorithm significantly outperform subsets selected by DTVNORM in all but two of the tested images for each metric. Figure 7.17 shows a detail for image 3, where by both measures DTVNORM outperformed the SBCS3 algorithm. The top row shows an area of the clouds where the canonical subset provided greater detail than the differential TV-norm subset. The bottom row shows the boat area, for which the canonical set gives a poorer reconstruction.

Figure 7.18 shows the reconstruction results for the images in Figure 7.14. The left column in each cell shows the original image, the second column shows the reconstructions from the top 200 points selected by differential TV-norm, the right column shows recon-

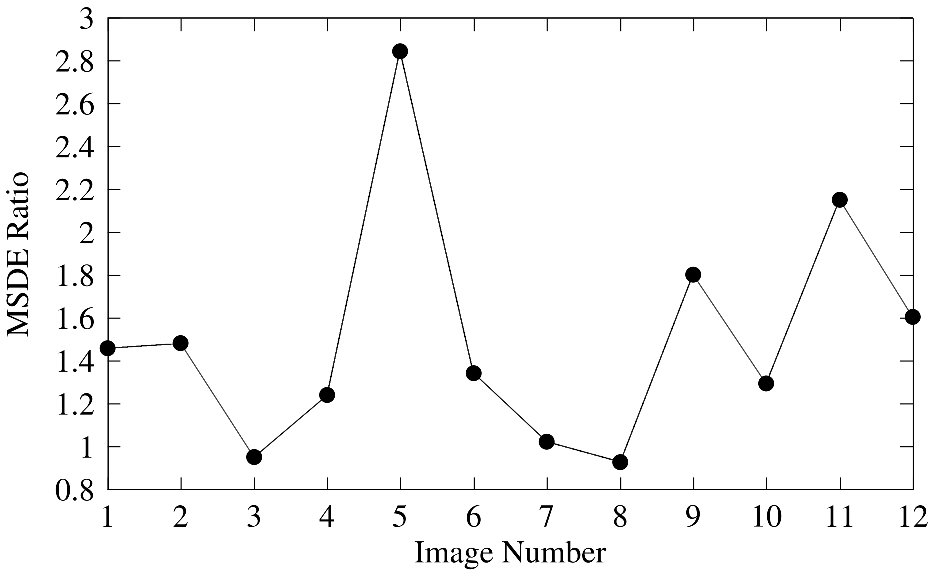


Figure 7.16: Ratio of MSDE measurements for DTVNORM/SBCS3 (values above 1 indicate the SBCS3 algorithm produces better reconstructions).

structions from 200 points selected using the canonical set algorithm. The reconstructions from the canonical sets of image points have lower error for all but two of the images. Further examination of the reconstructions shows reconstructions using the canonical sets avoid the large empty regions (areas with no selected features) present in the reconstructions using the ordered differential TV-norm.

Next the error rates for reconstructions using the top 200 points from the filtered combined interest point sets were inspected. Recall that the reduced sets are generated by discarding points within a radius of 3 pixels of one with a higher differential TV-norm. Figures 7.19 and 7.20 show the ratios for the SSIM and MSDE respectively. The SSIM ratios show the SBCS3 algorithm is better in 9 of the 12 reconstructions, while the MSDE ratios show the SBCS3 algorithm is better in 8 out of the 12 reconstructions and only slightly worse in 3 instances in terms of error rates. One reason for this behavior (being only slightly worse) is that for $\alpha = 0$, the canonical set algorithm is only considering the stability (differential TV-norm) of the points in the objective and the spreading of the points

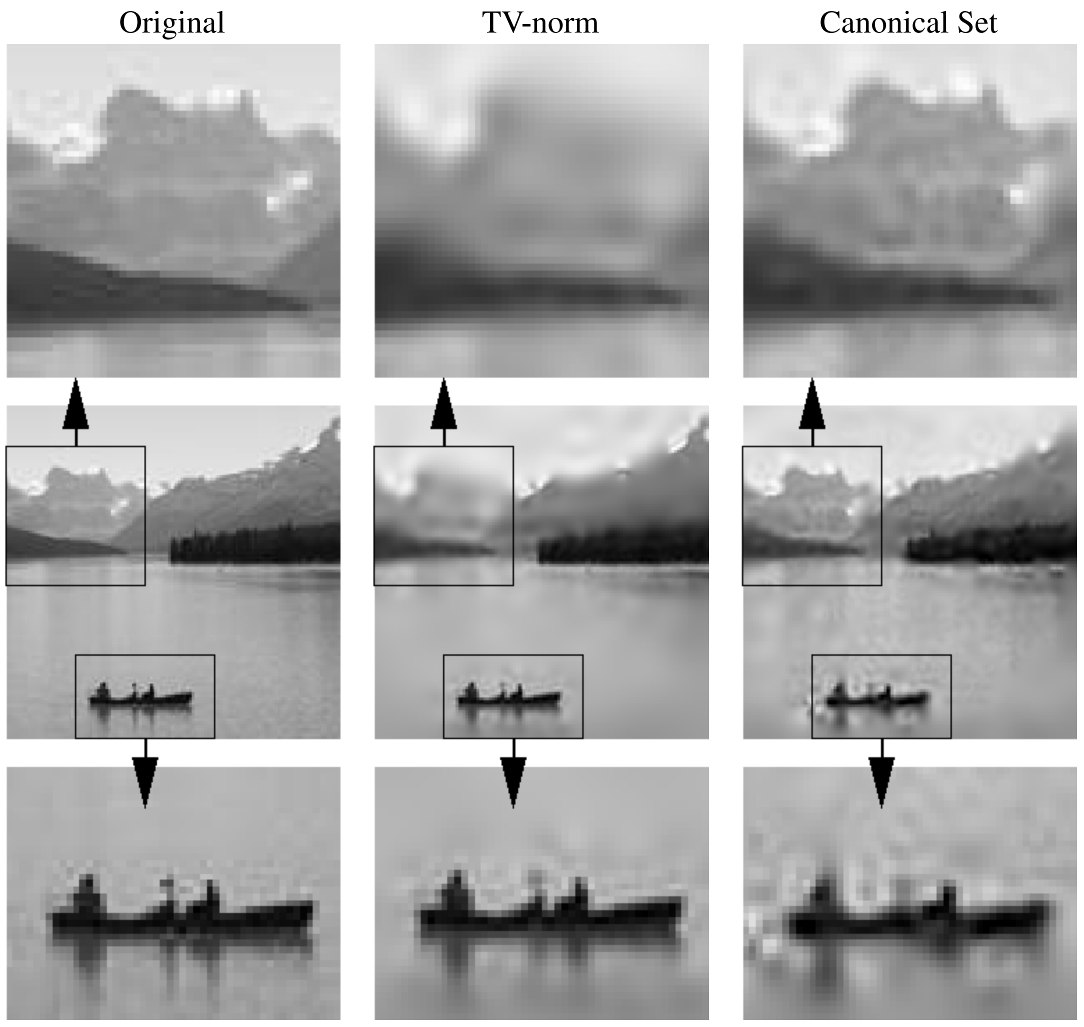


Figure 7.17: The left column shows the original image, the middle column shows the reconstruction using the differential TV-norm, and the right column shows a reconstruction using the canonical set. Even though the canonical set gives a better reconstruction of the clouds, the boat has reconstruction artifacts, resulting in a higher MSDE.

is enforced through the constraint (7.21).

Finally, the effect of the number of selected features on the quality of reconstruction was examined. Preliminary investigations indicated a progressive improvement on the quality of reconstruction as a function of subset size. Figure 7.21 shows the results of this investigation for image 9 from the test set, the MSDE measure gave similar results. Figure 7.22 shows reconstructions for various subset sizes.

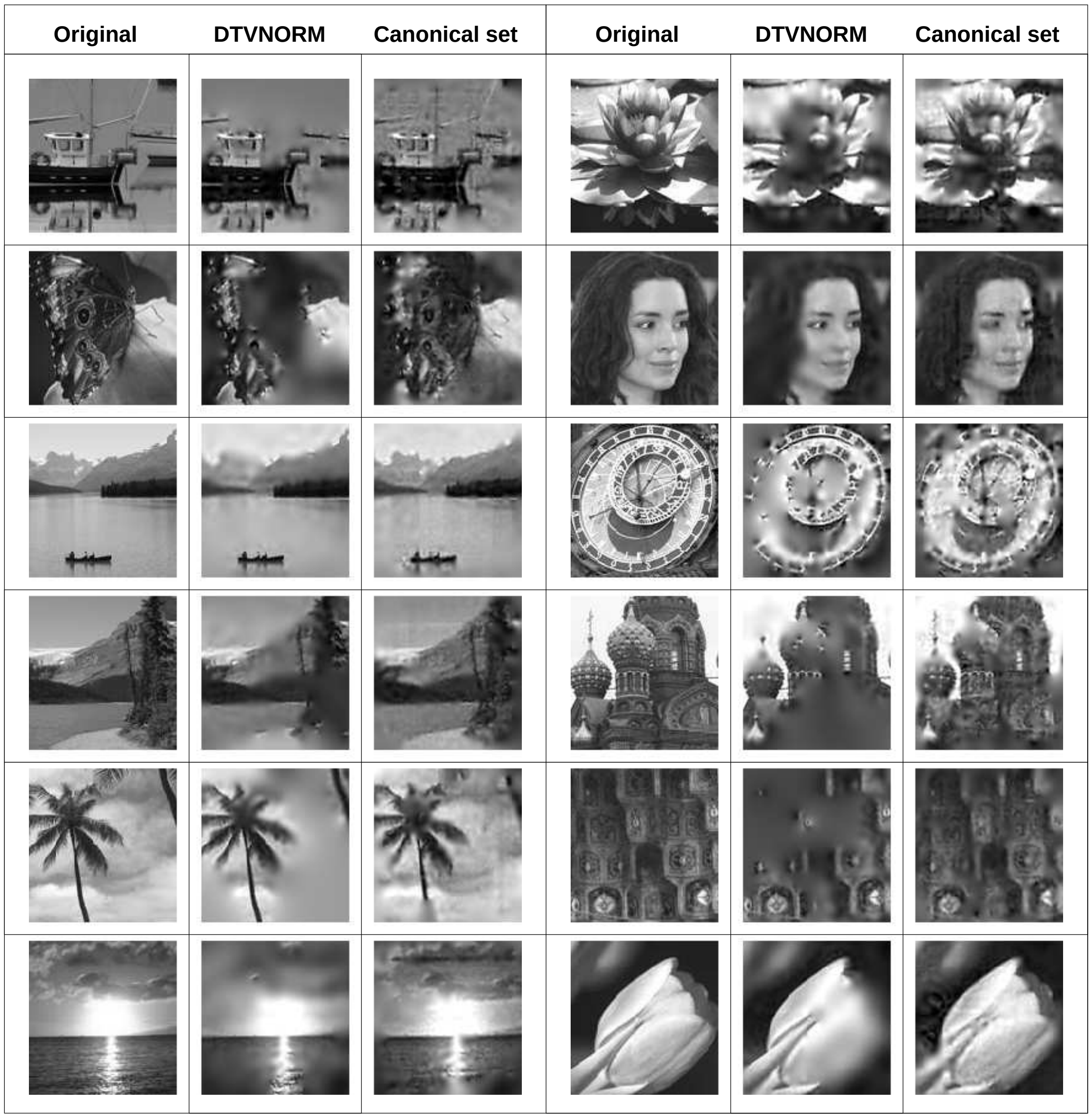


Figure 7.18: Image reconstructions, the left columns show the original images, the middle columns show reconstructions with the top 200 scale space points ordered by differential TV-norm, the right columns show reconstructions with the canonical set of scale space points computed by the SBCS3 algorithm.

7.6 Conclusion

This chapter has explored the use of the canonical set method for selecting a subset from a combined set of scale space interest points for image reconstruction. Section 7.2

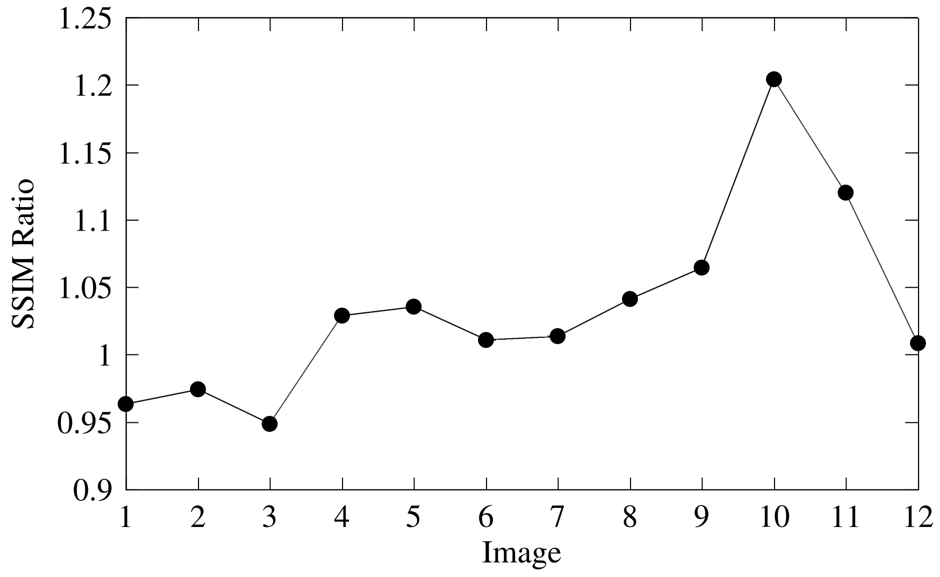


Figure 7.19: Ratio of SSIM measurements for SBCS3/min3 (values above 1 indicate the SBCS3 algorithm produces better reconstructions).

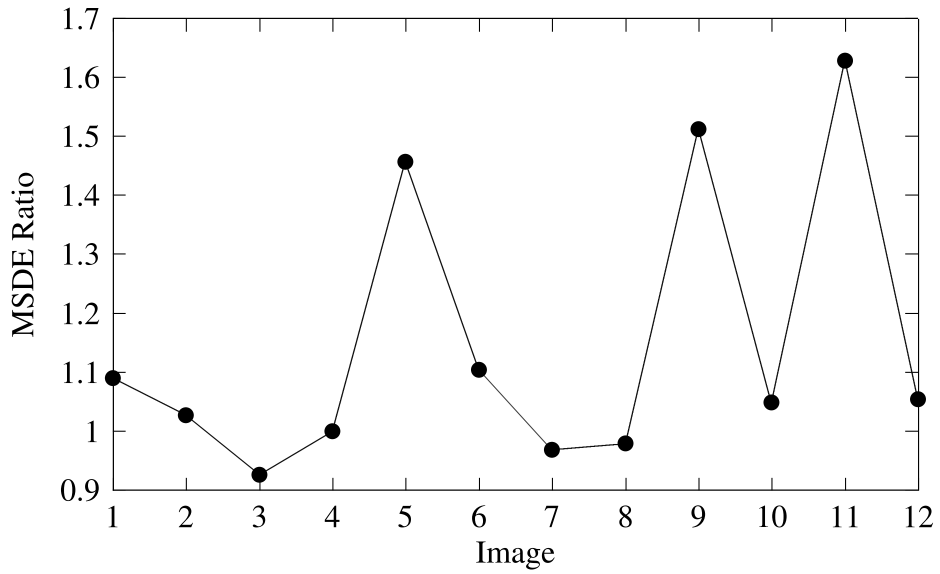


Figure 7.20: Error for top 200 min3 filtered versus best canonical set (lower is better). Ratio of MSDE measurements for min3/SBCS3 (values above 1 indicate the SBCS3 algorithm produces better reconstructions).

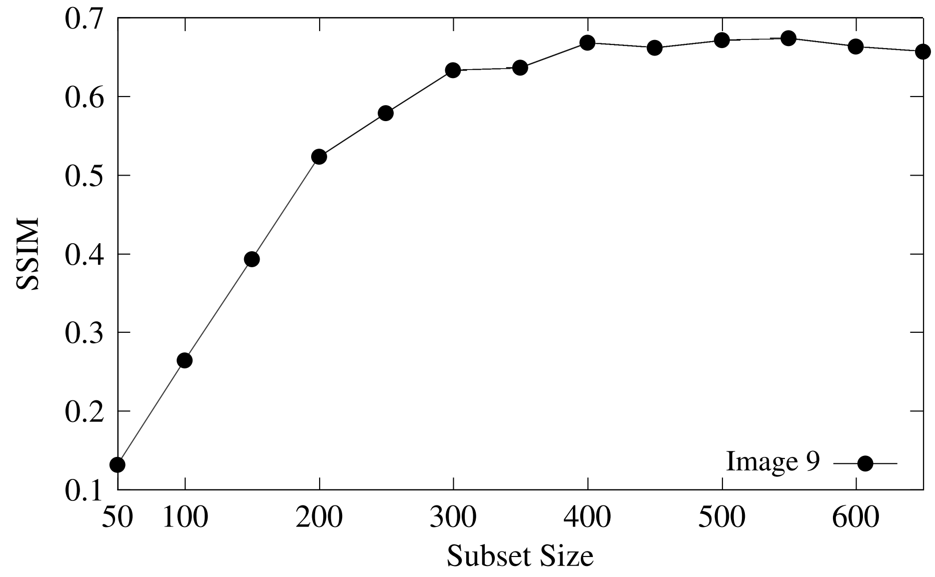


Figure 7.21: The effect of canonical subset size on the quality of reconstruction as measured by SSIM for image 9 from Figure 7.14. The reconstruction quality increases as the size of the subset increases.

introduced ten types of scale space interest points used in the experiments. Section 7.3 discussed reconstruction and selection based on the differential TV-norm. Then Section 7.3.2 described the two measures of reconstruction quality used in the experiments. Section 7.4 presented the algorithm for selecting subsets for combined interest points sets. Finally, Section 7.5 described experiments and presented results that evaluated the efficacy of the approach. The experiments showed the quality of the reconstructions based on the subsets of image features selected by the canonical set algorithm produced lower error as measured by the Multi Scale Differential Error (MSDE) in 10 of 12 reconstructions. In 8 of the 12 case studies the error was significantly lower, and in the other cases, they were comparable. In all cases the subsets selected using the SBCS3 algorithm avoided the large empty areas in images (areas with no selected features) that were present using the subsets selected by the ordered differential TV-norm.

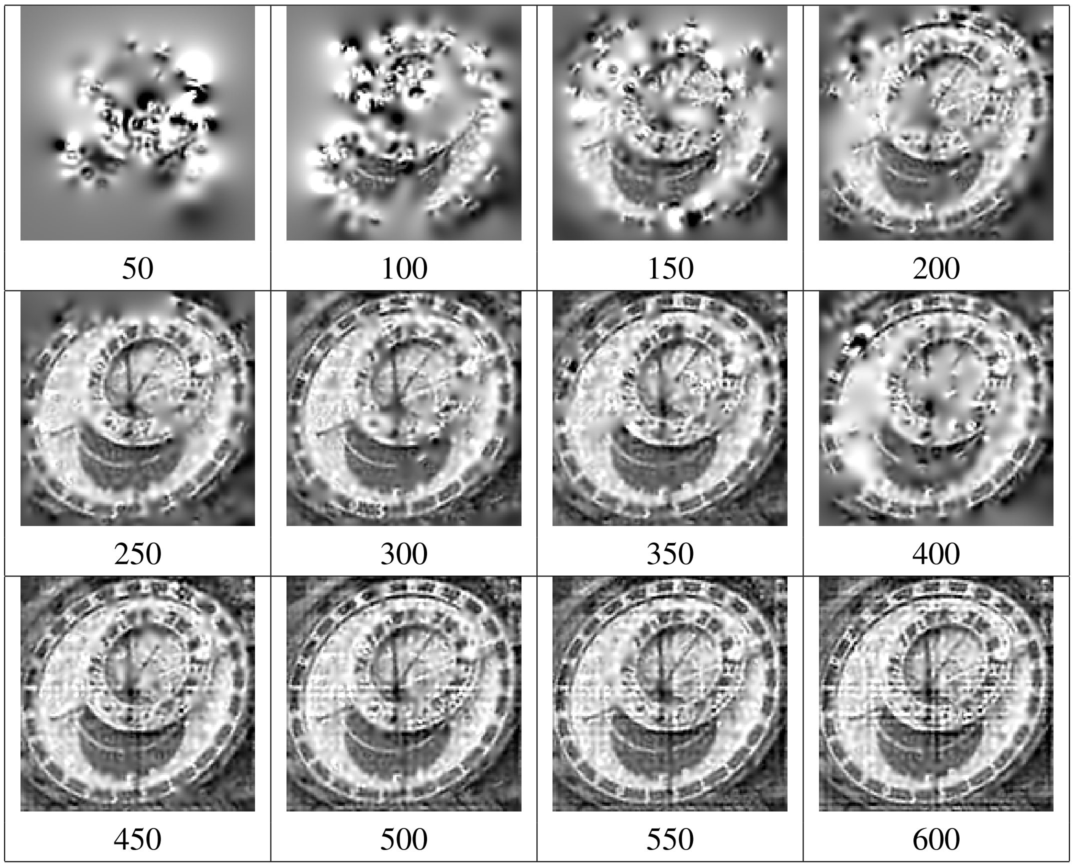


Figure 7.22: Reconstruction based on canonical sets of increasing size. The size of canonical sets for each reconstruction is indicated in the cells. The line artifacts are due to numerical errors in the reconstruction algorithm.

8. Canonical Set Applications in Software Engineering

8.1 Introduction

Although the canonical set algorithms were originally designed for computer vision applications, they have also been shown to be useful in other domains as well. In particular, the canonical set algorithms have played a key role in software engineering methods devoted to the understanding of software systems as shown by Kothari *et al.* [58, 59] and Salah *et al.* [98]. The ability of canonical sets to effectively summarize and represent a larger set facilitates the understanding of large complex software systems.

8.2 Usage Scenario Problem

Many software systems undergo continuous modification in response to changes in requirements, repairs of faults, and performance enhancements. To be effective, software engineers must understand how the software works before they attempt to modify it. Unfortunately, the size and complexity of the software often complicates the job of the software engineer, who may have to expend a significant amount of effort to comprehend the intricacies of the source code.

Documentation for object-oriented programs often includes descriptions of the parameters and return types of each method in a class, but little or no information on valid method invocation sequences. Knowing the sequence with which methods of a class can be invoked is useful for software engineers who are actively involved in the maintenance of large software systems.

The problem of knowing the valid method invocation sequences would be alleviated if every software interface had a specification, or at the very least a set of examples, that codified how its component could be used. Many programs have documentation (*e.g.*, Java programs have javadoc specifications) that describes the parameters and return types of each method in a class or the entry points of an API for a subsystem. Unfortunately, this type of documentation does not describe valid method invocation sequences.

In the event that class usage scenarios are unavailable, they can be generated by hand. This, however, requires intimate knowledge of the software system. Thus, it would be beneficial to be able to generate class usage scenarios automatically. This section describes Scenariographer [98], a tool for generating class usage scenarios, from method invocations, which are collected during the execution of the software. This algorithmic approach employs the canonical set method to categorize method sequences into groups of similar sequences, where each group represents a usage scenario for a given class.

8.2.1 Use of Canonical Set Method

Figure 8.1 illustrates the high-level architecture of Scenariographer, which consists of three major subsystems: data gathering, repository, and analysis. The data gathering subsystem collects runtime information, the repository subsystem stores runtime information collected by the data gathering subsystem, and the analysis subsystem identifies the various class usage scenarios.

The method sequence extractor (MS Extractor) component of the analysis subsystem extracts the method invocation sequences from the class instances and eliminates all self-calls (*i.e.*, invocations an object made to itself) from the method sequences. These sequences are then encoded as strings with a single character representing each distinct method call.

The MS Classifier (Method Sequence Classifier) component implements the BCS algorithm for approximating canonical sets. This algorithm is described in detail in Chapter 3. The input to the BCS algorithm consists of a similarity matrix. The matrix is constructed by computing the distance between every pair of method invocation sequences (strings)

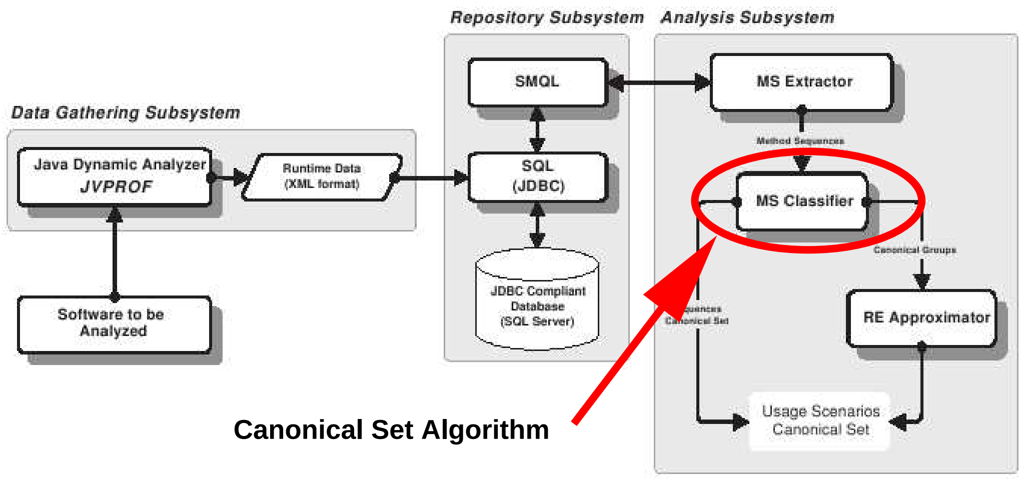


Figure 8.1: High-level architecture of Scenariographer with component implementing canonical set algorithm marked in red.

using the edit-distance [64]. Intuitively, the edit-distance between two method invocation sequences p_i and p_j measures the minimum cost of changes needed to transform p_i to p_j.

The MS Classifier uses the resulting canonical set members to create the canonical groups of method invocation sequences that correspond to class usage scenarios. The canonical groups are constructed by grouping each non-canonical element with the nearest canonical set member.

8.2.2 Scenariographer Results

The Scenariographer tool was used to generate usage scenarios for the widely-used class gnu.regexp.RE, which is used in the open source Java text editor Jext [45]. This class is part of the GNU regular expression package and provides an interface for compiling and matching regular expressions.

The Scenariographer tool produced three class usage scenarios, which can be approximated by the usage scenarios described in Table 8.1. Engineers who are not familiar with the gnu.regexp.RE class can consult these usage scenarios to gain insight into typical uses

Scenario 1	
`init`	Constructor called once.
`(match)+`	This method is called one or more times.

Scenario 2	
`init`	Constructor called once.
`(match | isMatch | chain)?`	Optionally, one of these methods is called.
`(isMatch)*`	This method is called zero or more times.

Scenario 3	
`init`	Constructor called once.
`chain`	This method is called once.
`(match)*`	This method is called zero or more times.

Table 8.1: Class usage scenarios generated for the gnu.regexp.RE class

for the class. Engineers can be confident that new code that corresponds to established usage scenarios will be unlikely to have unintended side effects.

8.3 Software Evolution Problem

Managers responsible for large software systems are always in search of techniques to measure and quantify the development trends in a project. These techniques help to ensure the long-term health of the software system and reduce the cost of maintenance. This section presents a technique that gives managers an overview of the code development process enabling them to characterize the major work activities during a time period. Finding trends in these work activities enables managers to better understand the software system's life-cycle and help them plan their development activities accordingly. The goal is to describe source code changes automatically. While it is possible to retrieve atomic changes in the code from source control repositories, this information is overwhelming and requires in depth knowledge of the system to comprehend.

The approach presented in this section examines the temporal evolution of code stored in source control repositories. It employs the notion of “canonical sets” to identify a subset of *Canonical Changes* that best represent the modification activities within a time period. These canonical changes act as centroids for the modifications applied to a system in a given time period, inducing a clustering. Using this clustering, the distribution of effort across the various change categories can be discovered. By studying the distribution of effort, managers can detect if the development team is spread thin focusing on too many areas or they are focused on a small number of tasks. The method produces these unbiased and statistical measurements automatically and permits the identification of canonical changes in partitions (periods) in the lifetime of long lived software projects.

8.3.1 Use of Canonical Set Method

Source control systems are used extensively by large software projects to control and manage their source code [96, 109]; examples are RCS [109], CVS [12, 26] and Perforce [92]. These systems help coordinate the development process between various members of the team and provide the ability to restore the source code to its state at any given time in the past.

The repository of a source control system usually tracks the creation, and initial content of each file. In addition, it maintains a record of every change to a file. For every change, a *modification record* stores the date of the change, the name of the developer who performed it, the specific lines that were changed (added or deleted), and a detailed explanation message entered by the developer giving the reason for the change. Using the information stored in the source control system, change sets (files that were changed together by the same developer within a short time frame) can be recovered. For selected time periods, change lists, which are the lists of all the changes applied to a system, can be recovered in the order that they were applied.

The first question is: What are the types of changes that were made during each period of the software's development? As compared to the work presented by Hassan and Holt [38], the approach presented here is based on the idea that the length of different periods need not be constant. The lifetime of a software system can be divided into successive periods of time as week, month, year, or any arbitrary time frame. The results presented in this section use periods of 3 months, where periods are defined as development up to the time in consideration.

Following partitioning of the development life cycle of the system into periods, the similarity between each pair of commits is computed using the Jaccard similarity measure. The Jaccard coefficient, $J(X, Y)$, is a measurement of asymmetric information on binary variables,

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

where X and Y represent individual commits. That is, they are sets of files representing a single commit. The similarity for each pair of commits is computed in order to obtain the similarity matrix for the SBCS approximation. In the context of change lists, this subset is referred to as the set of canonical changes, where the canonical changes represent the main types or clusters of changes.

The advantage of finding the canonical changes as opposed to a histogram of files represented in each change is that the canonical changes actually represent all the changes of the period that have been applied to the system. A histogram, on the other hand, simply measures the frequency of files in each change. Additionally, it is important to note that the number of changes applied to a system does not skew the results of the canonical changes, a very important feature of the approach.

8.3.2 Change Cluster Results

To demonstrate the feasibility of the method, the software life-cycle of the open source object-relational database, PostgreSQL was analyzed. The analysis of PostgreSQL begins with its first formal release, version 6.0.

Figure 8.2 shows the number of change clusters identified for each development period in the lifetime of PostgreSQL. The figure lists the major release number for each corresponding period. The number of change clusters reveals the varying amount of change activities during each time period. The canonical changes that form the basis of the change clusters are due to the inherent relationship between changes during the development process during each time period.

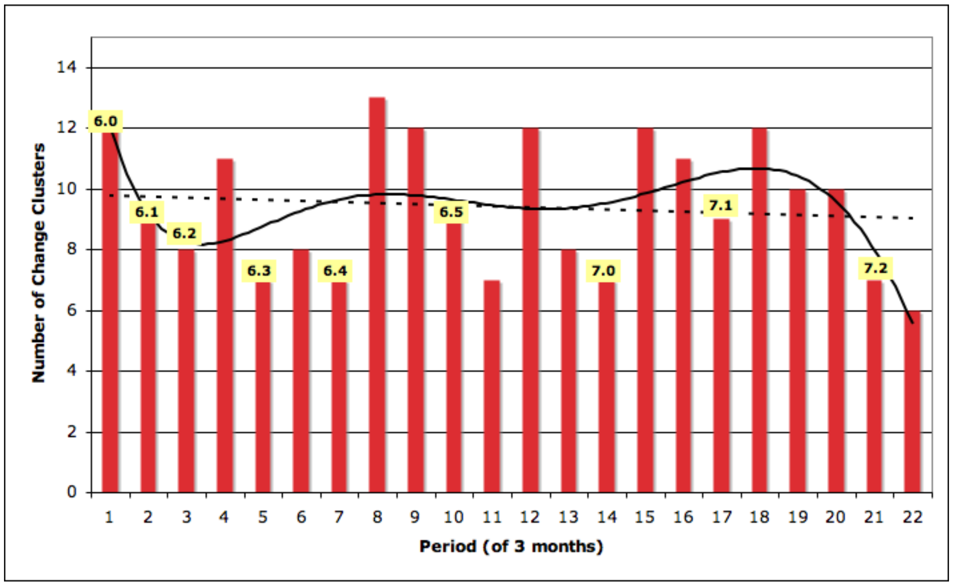


Figure 8.2: Number of identified change clusters of PostgreSQL by period

To uncover overall global trends in the results, linear and polynomial (of degree 6) trend lines were fitted to the data. The linear trend line (dotted line) shows a slight downward trend. This indicates that PostgreSQL's development activity has remained active over time at a reasonably constant rate. Examining Figure 8.2, one can observe that the polynomial trend line (solid line) shows a similar trend except toward the 7.2 release where there is a decline in the number of canonical clusters. This decline is likely due to missing change data for the 22nd period.

In addition to the global trends, Figure 8.2 shows a significant rise in the number of change clusters in several periods following a release when compared to the prior period. Moreover, note the decline in the number of change clusters in a period preceding a release, when compared to the prior period's. Between releases 7.0 and 7.1 or 7.1 and 7.2, there is an increase in the number of change clusters as development commences and a decline as development for the release winds down. This is probably because the development of the system began with a focus on several new areas, and as the release approaches, the focus shifts to fewer and fewer areas. Intuitively, this is expected since as goals for the release are accomplished, they are no longer worked on. As a release approaches, new work is not added, this is commonly known as a "feature freeze".

The approach of using canonical sets not only allows managers to determine what change activities are being focused on during a given period, but also provides more information. Specifically, it provides managers with information about the number of areas where changes have been applied. This allows them to see when the development team has focused on several activities or just a few. It also shows managers what those activities are by providing representative examples of them.

This method also allows identification of when code maintenance or refactoring work is being performed as compared to new development on the software system. Comparing the canonical changes of two consecutive periods provides information that reveals how the focus of development changes from one period to the next. Further details on this method are described by Kothari *et al.* [59].

8.4 Software Features Problem

One way of understanding a software system is to identify and comprehend its features and the code that implements those features. For complex applications the number of features may be significant, and indeed overwhelming. The implication is, that in order to understand a software system, all of its features must be understood. However, it is likely that there are similarities between features. Therefore, understanding one feature assists in understanding similar features.

Using a measure of similarity between pairs of software features, the set of canonical features of a software system can be obtained. The canonical features set (CFS) consists of a small number of features, relative to the total number of features, that are most representative of the software system. Intuitively, the CFS is a set of distinguished features that characterizes a software system succinctly.

Engineers can obtain an overview of a system's capabilities by studying the features in its CFS. This follows from the fact that all other features of the system are similar to features in the CFS. Because members of the CFS are as dissimilar as possible, engineers can obtain a broad overview of the system by inspecting a small number of its features.

Engineers can also obtain an overview of a system's implementation by studying how members of its CFS have been implemented. Two features that have similar functionality should share code and thus only one of the two features should appear in the CFS. Transitively, if many features are similar to each other, as expected in a large system where features share code, only one of those features should appear in the CFS.

The features in the CFS act as central points for all of the features, and induces a clustering that partitions the features of a software system. Each of the clusters is anchored by a canonical feature and contains feature elements that are similar to the canonical feature.

Partitioning the set of features in this manner can reduce the cost of understanding large and complex software systems. When an engineer knows that a specific feature must undergo maintenance, understanding which features are most closely related is helpful since related features most often share significant portions of their code. This method leverages the engineer's time investment by presenting a minimal number of specific features that reveal the most about the system.

8.4.1 Use of Canonical Sets

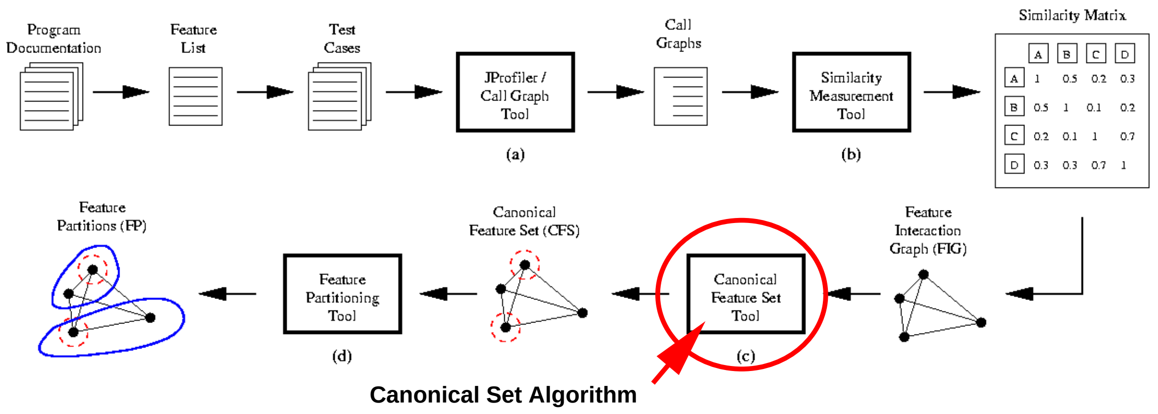


Figure 8.3: Work-flow and tool chain; a) Call graph tool, b) Similarity measurement tool, c) Canonical feature set tool, d) Feature partitioning tool. The canonical set algorithm is implemented in component (c).

Figure 8.3 depicts the work-flow and tool chain that describes the process of computing Canonical Feature Sets (CFSs) and Feature Partitions (FPs) of software systems. First the features of a software system are identified using documentation, use cases, test cases, or other means such as the help system of the software. Next, a set of test cases is designed to exercise the features of the program. The features are executed under the supervision of a dynamic analysis tool that captures the objects, functions, and variables that were involved. If a test suite is present for the software, the test cases corresponding to individual features may be utilized, rather than manually exercising the features.

The test cases are used with the call graph tool (Figure 8.3a) to produce a set of call graphs, one for each executed feature of the software being studied. Using the similarity measurement tool (Figure 8.3b), the pairwise similarity between the call graphs is computed. Since call graphs are a direct representation of feature implementation, the similarity between two call graphs is equivalent to the similarity of the features being represented by those call graphs. In order to determine the similarity between two features, the association graph based on the call graphs of those features is computed. The similarity can then be defined as the cardinality of the maximal clique in the computed association graph [91].

The CFS tool (Figure 8.3c), uses the matrix of pairwise similarities between features to determine the canonical feature set using the SBCS algorithm described in Chapter 3. Lastly, the feature partitioning tool (Figure 8.3d), is used to cluster the remaining, non-canonical features, using the canonical features as partition representatives. To partition the features the technique places every non-canonical feature in a set with its nearest neighbor in the CFS. Clusters thus represent sets of similar features based on implementation.

8.4.2 Firefox Suite Results

This technique was applied to the Mozilla-based web-browser, Firefox, and its companion mail and news client, Thunderbird. The browser provides features such as an integrated pop-up blocker, tabbed browsing, built-in search, live bookmarks, themes, and the ability to apply extensions to the browser for added custom functionality [24]. Since Firefox and Thunderbird share a code-base, they were treated as a single integrated system.

The versions of Firefox and Thunderbird that were considered in this analysis were 1.0.6 and 1.0.7 respectively, the two latest releases at the time. To gauge the size of the system, and implicitly the complexity of its source code, note that the Firefox suite is implemented using over 3 million lines of code and 10,000 source files, not including the shared libraries. The task of understanding a system of this size by examining its code is daunting. After computing the pairwise similarity between all of the use-cases, the Canonical Features Set was computed. The CFS for the Firefox suite can be seen in Table 8.2. Interestingly, the CFS does not explicitly state that browsing is a canonical feature of the

	Feature Name
1	File-Open Location
2	Bookmark-Add
3	Get Mail
4	Send Link
5	Edit-Find in This Page

Table 8.2: CFS of Firefox/Thunderbird Suite

suite. However, inspection of the similarities of features show that `Open Location in Tab`, `Open Location in Window`, and `Go to Location` are all very similar to `File-Open Location`. Any of these features could represent the group of features containing those, and similar features since they all represent the browsing functionality.

Justification as to why certain features appear in the CFS can be found in the release notes of the Firefox and Thunderbird applications, as well as their implementations. The first surprising result is that `Send Link` is in the CFS and `Send Message` is not. This is because `Send Link` subsumes the functionality of `Send Message` as well as places the link to be sent in the message being written. `Send Link` opens an email composition containing the link as the body of the message. In actuality, the `Send Link` feature invokes a mailer command that does not necessarily need to be handled by Thunderbird, but by the default mail client. Similarly, `Left Click on email address` executes the same functionality of creating an email message within the Firefox suite.

This work contributes to the state-of-the-practice in software engineering by providing software engineers with tools that can assist them in either a broad or targeted study of a software system. By examining each canonical feature and the classification of features, software engineers can get a broad overview of the distinguishing features of the software. This is especially helpful in the absence of high-quality software documentation. Similarly, by examining a cluster of the partitioned feature set and by examining the similarity between the features in the same cluster, the software engineer can perform a systematic and targeted study of a distinct set of software capabilities.

8.5 Characterizing Software Evolution

As shown in the previous sections in this chapter, a current focus of software engineering research in on the development of techniques to measure and quantify the development trends in a software project. Section 8.4 described a technique that automatically identifies the canonical features of a software application. The canonical features are relatively few in number when compared to the total number of features of an application. By understanding the implementation of these canonical features of an application an engineer can obtain an informative abstraction of the implementation of the entire application. This result makes progress toward reducing the cost of understanding and maintaining similar features.

The hypothesis examined in this section is that through continual refactoring, the implementations of the features in the clusters will continually become more similar to each other. That is, the implementations of features in a feature cluster will *converge* and their similarity will increase over time. Conversely, if opportunities for refactoring were overlooked, it is suspected that the implementations of semantically similar features may not converge over time. Through the measurement of this feature convergence, the development process may be characterized.

Feature implementation overlap is the degree of similarity between two features. As in Section 8.4.1, the similarity is the cardinality of the maximal clique in the computed association graph [91]. This measure takes into account the number of shared method calls as well as the inherent structure of the calls made (*i.e.*, call graph). Features that have similar call graphs, have a higher implementation overlap.

Intuitively, one can think of overlap as a more strict similarity measure as compared to method reuse which only consists of the nodes of the call graph. Features exhibiting significant overlap have a high degree of method reuse; since, not only are the same methods being used between features, but also in the same pattern. Alternatively, it is possible to observe a low degree of overlap and a high degree of method reuse. This indicates that although the same methods are being used between two features, they are not being used in the same way.

The convergence of a feature with regards to the inherent structure of the system is defined as:

$$Convergence(f_k) = \frac{\epsilon(f_k, C(f_k))}{avg(\epsilon(f_k, \{C_*\} - C_{f_k}))} \tag{8.1}$$

where f_k is a specific feature, $\{C_*\}$ indicates the set of all canonical features, C_{f_k} is the canonical feature with the greatest implementation overlap with f_k (*i.e.,* its canonical feature), and $\epsilon(f_k,C_{f_k})$ is the degree of overlap of f_k with its canonical feature. The term $avg(\epsilon(f_k,\{C_*\} - C_{f_k}))$ is the average overlap of feature f_k with all the canonical features except its own canonical feature. In other words, it measures the ratio of the degree of overlap between a single feature and its canonical feature; and the average of the overlap of that same feature with all of the other canonical features.

8.5.1 Use of Canonical Sets

The tool chain shown in Figure 8.4 depicts the process of evaluating the convergence of an application, as well as the intermediate results of the various tools used. The CFS tool (8.4b.4) uses the SBCS algorithm described in Chapter 3 to compute the approximate canonical set. The Feature Clustering Tool (8.4b.5) then assigns each of the non-canonical features to a cluster represented by the nearest canonical feature.

8.5.2 Gaim Results

This technique was applied the open-source instant messaging client, Gaim. The versions of the system were obtained from the subversion repository listed on the application's

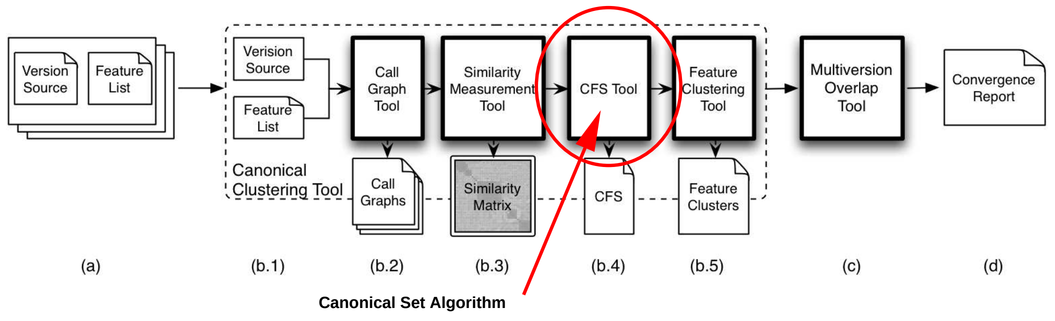


Figure 8.4: Tool chain; a) Multiple versions of a software system with corresponding feature lists; b) Canonical clustering tool which finds the canonical features, and clusters the features around the canonical feature set; c) Multiversion overlap tool calculates the overlap in features over time, d) Convergence report which provides an evaluation of the convergence of the implementation of the features of a cluster.

web page. For each version, all the features of the application were listed, and call graphs of their execution were obtained. Using these call graphs, the implementation overlap of all pairs of features for each version of the application was computed. Then for each version the canonical subset of features was computed. Each version had approximately 80 features that were reduced to a subset of six canonical features. Table 8.3 lists the canonical features of Gaim version 1.1.

	Feature Name
1	Send Message MSN*
2	Send File MSN*
3	View Log
4	New Away Message AIM*
5	Add Buddy AIM*
6	Set User Info

Table 8.3: CFS of Gaim Instant Messaging Client. The CFS was obtained using all the features of Gaim together. An * indicated that this feature is repeated for different protocols.

Although the features of Gaim exhibit a high level of method reuse, they do not exhibit a consistently high level of implementation overlap. The cases where implementation overlap is high can be attributed to the use of the graphical user interface (GUI) in the software system, and the repetitions of features for different protocols. For example, the same feature `Send Message` is repeated for all the messaging protocols that Gaim supports, such as Oscar, Yahoo!, and MSN. There is a separate version of the `Send Message` feature for each protocol. The implementations for each of these versions however, do share the same GUI code, and therefore have high similarity and implementation overlap since the GUI code constitutes a large portion of the code and is identical.

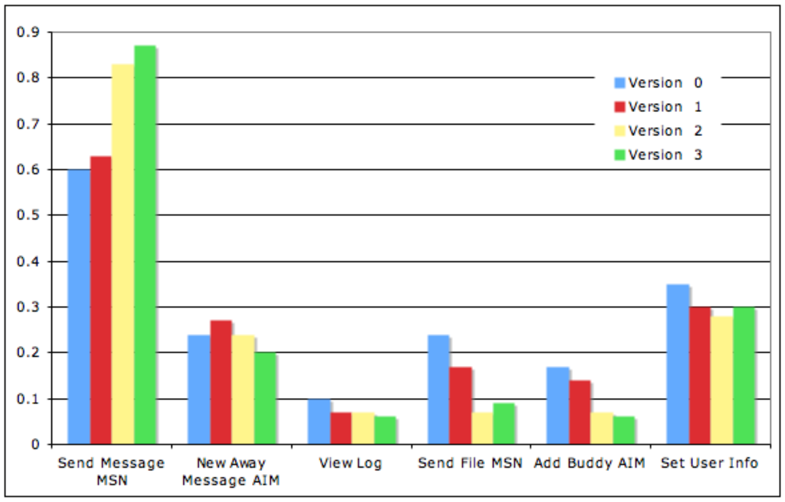


Figure 8.5: Implementation overlap of feature `Send Message AIM` with the canonical features of Gaim for four different versions of the system.

Quantitatively analyzing the convergence and implementation overlap of the `Send Message AIM` feature using Equation 8.1 reveals a steady increase in the convergence, as can be seen in Figure 8.5. The measure is 3.32, 3.85, 5.69, and 6.13 for versions 0 through 4, respectively. Observe the convergence of the feature is constantly and steadily increasing. Similarly, the measure of convergence exhibits the same trend for the majority of features of Gaim. Averaging the convergence measures for all features of the system for each version reveals a steady monotonic increase in this value, indicating that the system is becoming more stable. The features of Gaim have steadily been refactored. The features of the system are split into two parts each; the common part of the feature, and the protocol specific part. The usage of the methods are nearly identical for all the clusters of features, except for the protocol specific portions of the code. The developers of the messaging client practiced information hiding effectively. If the specifications of any of these messaging protocols were to change, the other protocols would not be affected. Furthermore, if they decide that they want to modify the messaging feature of the application, they only need to do so in one central place.

Studying Gaim, the effectiveness of the approach has been demonstrated, and the results can be justified based on the development of Gaim. This method contributes to the state-of-the-art in software understanding research by providing a measure of the convergence of software features over time, and demonstrating its effectiveness in characterizing the development of software systems.

8.6 Conclusion

This chapter has presented applications of the canonical set method in the field of software engineering. Section 8.2 showed how the canonical set method can be used to select representative class usage scenarios. Engineers can use these scenarios to gain insight into typical uses for the class. Section 8.3 showed how the canonical set method can be used to extract canonical changes from source code repositories. These canonical changes best represent the modification activities within a time period, thus aiding in the understanding of software development projects. Then Section 8.4 showed how the canonical features of a software system can be found. Understanding the canonical features of a software system can give engineers an overview of a system's capabilities without the need to examine all of the code. Finally, in Section 8.5, how features in the canonical feature clusters converge over time was studied. Examining how the features converge provides insight into the health of the software development process.

9. Conclusions

9.1 Summary

This thesis has explored the problem of representing a large dataset with a smaller more compact form through the use of nonlinear optimization methods. The problem of representing a large dataset with a smaller more compact form can be accomplished either by synthesis of new data points to represent clusters of the data points or by selecting a subset of the data points to represent the original data set. The approach presented here selects a highly representative subset of data points called a canonical subset.

Canonical subsets are constructed by formulating the subset selection problem in graph theoretic terms and then defining subsets with well-defined properties based on graph metrics. Formulations were given for three flavors of canonical sets in terms of mixed quadratic integer programs.

The first variation, the MDMC canonical subset, is the minimum dominating set with maximum cut-weight. Intuitively, such a set is a simultaneous solution to the minimum dominating set and the maximum cut problems in a graph. This canonical subset has the following attributes:

1. Data points in the canonical set are maximally similar to data points not in the canonical set.
2. The size of the canonical set is as small as possible.
3. Every data point is either in the canonical set or is similar to a data point in the canonical set.

The motivation for the second variation came from the observation that being able to specify the size of the canonical set would be useful. In addition, it might be desirable to have the canonical set members be as dissimilar as possible. The second variation then, the bounded canonical set (BCS), selects a subset with the following attributes:

1. Data points in the canonical set are minimally similar.
2. Data points in the canonical set are maximally similar to data points not in the canonical set.
3. The size of the canonical set is as least k_{min} and at most k_{max}.

The third variation, the stable bounded canonical set (SBCS), is a generalization of the BCS with the additional attribute that data points in the canonical set are maximally stable. The stability is a scalar value associated with each data point that is application specific. The motivation for developing this variation stemmed from the observation that in some cases additional information is available about the data points, *e.g.* the strength of an image feature.

These three canonical set problems are shown to be NP-hard, thus motivating the need for approximation algorithms. To obtain approximate solutions to the NP-Hard canonical set problems, a framework was developed which may be used to describe the canonical sets in terms of semidefinite programming optimizations. The SDP optimizations can then be relaxed and approximate solutions obtained. Performance proofs on the SDP approximations were then presented. An alternative method of approximation, quadratic programming, was then employed to formulate the canonical set problems as QP approximations.

The framework that was developed for the SDP and QP approximation methods was presented in such a way as to allow researchers to design their own canonical sets. That is, tables of potential objective functions were presented that would allow an algorithm designer to mix and match desired properties and construct approximation algorithms in a simple manner.

One question of interest, and, indeed, a motivation for design of the canonical set algorithms, is the following: Can subset selection be done with less sensitivity to outliers than traditional methods such as K-means? To explore this question, a small set of experiments were presented which showed that at least in some collinear point sets, canonical subset are more resistant to outliers than subsets selected using K-means.

The canonical set algorithms were originally designed for computer vision applications such as 2D view selection. The objective in this case is to identify highly informative 2D views of a 3D object. These canonical views can then be used in technical drawings, computer visualizations, and for 3D object recognition. Through two sets of experiments canonical subsets of 2D views were shown to well represent the 3D objects from which the views were taken.

Another application in the computer vision domain for the canonical set method is in selection of subsets of image features for the purpose of object localization. Image features are interesting portions of an image that conform to some statistical measure for the purposes of understanding or analyzing the image. Object localization is the process of determining the position, orientation, and scale of a query object (model) in a target scene. An extensive set of experiments was presented showing subsets of image features obtained using the canonical set method may be used to localize occluded objects in synthetic scenes. To objectively measure the performance of the algorithm, which was also compared with other existing subset selection techniques, a database of synthetic scenes of occluded objects with ground truth was constructed. Additionally, results on real scenes of occluded objects were presented.

Scale space interest points are image features that encode the deep structure of the input images. The increasing use of interest points in applications such as recognition, classification, and image editing has instigated research on interest points and their descriptive power. One way of evaluating the information content of interest points is through image reconstruction. To improve the accuracy of the reconstructions different types of interest points may be used, as they encode different aspects of the input image. However many of the combined interest points encode redundant information, thus motivating the selection of a subset of the combined interest points.

The extensibility of the canonical set framework was demonstrated through incorporation of spatial constraints into the scale space interest point subset selection process. A proof of how the optimal spatial constraints can be computed in polynomial time was presented and results from a comprehensive set of experiments on a suite of test images were presented. These experiments showed that a modified SBCS algorithm incorporating spatial constraints could effectively select subsets of different types of scale space interest points that resulted in reconstructions of higher accuracy.

Although the canonical set algorithms were originally designed for computer vision applications, they are general in nature. As proof of their generality, results of Kothari *et al.* [58, 59] and Salah *et al.* [98] were presented that show that the canonical set method is also useful in the field of software engineering. In particular, the canonical set method was shown to be useful for class usage scenario generation, studying the temporal evolution of code stored in source control repositories, and characterizing the evolution of a software system using the canonical features of the software.

9.2 Contributions

The contributions of this thesis are both theoretical and experimental. The experimental results validate the theoretical contributions by demonstrating the applicability of the canonical set method to different problem domains.

9.2.1 Theoretical Contributions

The theoretical contributions of this thesis are twofold. First, a new characterization of the subset selection problem as an optimization problem was presented. This is important, because subset selection is an important problem in many application domains. By characterizing subset selection as an optimization problem, comprehensive formulations of the problem become possible. These problem formulations were then subsequently proven to be NP-hard.

The second theoretical contribution of this thesis is the development of approximation methods that can compute canonical subsets. The algorithmic complexity of this framework was investigated and bounds on the quality of the solutions were provided. This is important because a practical way to compute approximate solutions to the NP-hard formulations allows the use of the algorithms in practical situations.

9.2.2 Experimental Contributions

The utility of the canonical set approximations were demonstrated in a series of experiments where the algorithms were used to select small sets of 2D views of 3D objects. Two sets of experiments showed that the canonical subsets of 2D views well represented the 3D object from which they were taken. These results are significant because 2D view selection is an important problem in computer vision, as discussed by Bowyer and Dyer [6] and others [76, 89].

Next, a comprehensive set of experiments showed that canonical subsets of image features could be used for the task of object localization under occlusion. The results showed that canonical subsets of image features outperformed subsets selected by conventional techniques, such as K-Means clustering, in the number of detections while at the same time maintaining higher accuracy of the localization. These experimental results are significant for applications where reducing the size of an object's representations is desirable, as shown by the work of Heisele *et al.* [39], Sivic *et al.* [105], and Sun *et al.* [108].

A further contribution was made in the field of computer vision on the problem of image reconstruction. A new method of selecting a subset from a combined set of scale space interest points for image reconstruction was presented along with a proof of how the optimal spatial constraints can be computed in polynomial time. These results are significant, and extend the work of Kanters *et al.* [49] and others [93].

Finally, the results of Kothari *et al.* [58, 59] and Salah *et al.* [98] showed that the canonical set method is useful in software engineering research. These results also indicate the canonical set method is a general method that can be used in problem domains outside of computer vision.

9.3 Future Work

The canonical set properties listed in Table 2.1 suggest that other combinations of properties might produce subsets of interest. For instance, maximizing the sum of the intra edges (the similarity of the data points in the canonical set) while minimizing the sum of the cut edges (the similarity of canonical to non-canonical data points) might be an effective way of extracting tight clusters from a dataset. Extending the lists of properties and constraints also seems a fruitful direction for study.

Finding the best weighting values for the multi-objective formulations requires experimentation. Perhaps some way could be found to predict what the best weights might be, or at least some heuristics to guide the researcher.

A thorough exploration of the effects of different types of similarity measures would also be interesting. Such a study could examine different types of distance measures on an extensive set of synthetic graphs. This could lead to a deeper understanding of canonical sets. For example, how does the canonical set change when different L_p norms or non-metric distance measures are used.

The existence of bounds on the semidefinite approximations and absence of bounds on the quadratic approximations suggests that it would be useful to have additional data and comparisons between the approximation methods. Perhaps some criteria could be developed that could predict which type of algorithm would be most suitable for a given application.

Bibliography

[1] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.

[2] Xiao Bai, Hang Yu, and Edwin R. Hancock. Graph matching using spectral embedding and semidefinite programming. In *Proceedings, British Machine Vision Conference (BMVC '04)*, September 2004.

[3] R. Bar-Yehuda and S. Moran. On approximation problems related to the independent set and vertex cover problems. *Discrete Applied Mathematics*, 9:1–10, 1984.

[4] Stefan Berchtold, Christian Böhm, and Hans-Peter Kriegel. The pyramid-tree: Breaking the curse of dimensionality. In Laura M. Haas and Ashutosh Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 142–153. ACM Press, 1998.

[5] A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997.

[6] K. Bowyer and C. Dyer. Aspect graphs: an introduction and survey of recent results. *IJIST*, 2:315–328, 1990.

[7] L. Bretzner and T. Lindeberg. Qualitative multi-scale feature hierarchies for object tracking. In *Journal of Visual Communication and Image Representation*, volume 11, pages 115–129, 2000.

[8] Matthew Brown, Richard Szeliski, and Simon Winder. Multi-image matching using multi-scale oriented patches. In *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition (CVPR05)*, volume 1, pages 510–517, 2005.

[9] T. Brox and J. Weickert. A tv flow based local scale measure for texture discrimination. In T. Pajdla and J. Matas, editors, *Proc. 8th European Conference on Computer Vision, Prague, Czech Republic.*, volume 2 of *Computer Vision - ECCV*, pages 578–590. Springer LNCS 3022, May 2004.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, second edition, 2003.

[11] T. Cover and J. Thomas. *Elements of Information Theory: Rate Distortion Theory*. John Wiley & Sons, 1991.

[12] CVS - Concurrent Versions System. Available online at `http://www.cvshome.org`.

[13] C. M. Cyr and B. Kimia. 3d object recognition using shape similarity-based aspect graph. In 8^{th} *Inter. Conf. Comp. Vision*, pages 254–261, 2001.

[14] M. F. Demirci, A. Shokoufandeh, S. Dickinson, Y. Keselman, and L. Bretzner. Many-to-many graph feature matching using spherical coding of directed graphs. In *ECCV*, May 2004.

[15] M. Fatih Demirci, Ali Shokoufandeh, Yakov Keselman, Lars Bretzner, and Sven Dickinson. Object recognition as many-to-many feature matching. *IJCV*, 69(2):203–222, 2006.

[16] T. Denton, J. Novatnack, and A. Shokoufandeh. Drexel object occlusion repository (DOOR). Technical Report DU-CS-05-08, Drexel University, Computer Science Department, 2005.

[17] Trip Denton, Jeff Abrahamson, and Ali Shokoufandeh. Approximation of canonical sets and their application to 2d view simplification. In *CVPR*, volume 2, pages 550–557, June 2004.

[18] Trip Denton, M. Fatih Demirci, Jeff Abrahamson, Ali Shokoufandeh, and Sven Dickinson. Selecting canonical views for view-based 3-d object recognition. In *ICPR*, pages 273–276, August 2004.

[19] S. Dickinson, A. Pentland, and A. Rosenfeld. A representation for qualitative 3-D object recognition integrating object-centered and viewer-centered models. Technical Report CAR-TR-453, Center for Automation Research, University of Maryland, 1989, (also appears with the same title in: K.N. Leibovic (ed.), *Vision: A Convergence of Disciplines*, Springer Verlag, NY, 1990, pp 398–421.).

[20] D. Eggert and K. Bowyer. Computing the orthographic projection aspect graph of solids of revolution. *Pattern Recognition Letters*, 11:751–763, 1990.

[21] D. Eggert, K. Bowyer, C. Dyer, H. Christensen, and D. Goldgof. The scale space aspect graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1114–1130, November 1993.

[22] Matthias Ehrgott. *Multicriteria Optimization*, volume 491 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, New York, 2000.

[23] Uriel Feige, Marek Karpinski, and Michael Langberg. A note on approximating Max-Bisection on regular graphs. *Information Processing Letters*, 79(4):181–188, 2001.

[24] Firefox - Rediscover the Web, Firefox Homepage. http://www.mozilla.com/firefox. May, 2006.

[25] L. Florack and A. Kuijper. The topological structure of scale-space images. *Journal of Mathematical Imaging and Vision*, 12(1):65–79, February 2000.

[26] K. Fogel. *Open Source Development with CVS*. Coriolos Open Press, Scottsdale, AZ, 1999.

[27] Robert M. Freund. Introduction to semidefinite programming, 2004. Lecture 23 http://ocw.mit.edu/OcwWeb/Sloan-School-of-Management/15-084JSpring2004/CourseHome/index.htm.

[28] A. Frieze and M. Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. In *Integer Programming and Combinatorial Optimization*, volume 920, pages 1–13. Springer, 1995.

[29] M. X. Goemans. Semidefinite programming in combinatorial optimization. *Mathematical Programming*, 79:143–161, 1997.

[30] Michel X. Goemans and David P. Williamson. .878-approximation algorithms for max cut and max 2sat. In *Twenty-sixth Annual ACM Symposium on Theory of Computing*, pages 422–431, New York, 1994.

[31] Michel X. Goemans and David P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. Assoc. Comput. Mach.*, 42:1115–1145, 1995.

[32] G. Golub and C.V. Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 1996.

[33] S. Gordon, H. Greenspan, and J. Goldberger. Applying the information bottleneck principle to unsupervised clustering of discrete and continuous image representations. In *Proceedings, International Conference on Computer Vision*, Nice, France, October 2003.

[34] Eran Halperin and Uri Zwick. A unified framework for obtaining improved approximation algorithms for maximum graph bisection problems. *Lecture Notes in Computer Science*, 2081:210+, 2001.

[35] R.M. Haralick. Ridges and valleys on digital images. *Computer Vision, Graphics and Image Processing*, 22:28–38, 1983.

[36] C. Harris and M. Stephens. A combined corner and edge detector. In *4th ALVEY vision conference*, pages 147 – 151, 1988.

[37] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. Alvey Vision Conf., Univ. Manchester*, pages 147–151, 1988.

[38] Ahmed E. Hassan and Richard C. Holt. The Chaos of Software Development. In *Proceedings of the 6th IEEE International Workshop on Principles of Software Evolution*, Helsinki, Finland, September 2003.

[39] B. Heisele, T. Serre, S.Mukherjee, and T.Poggio. Feature reduction and hierarchy of classifiers for fast object detection in video images. In *Computer Vision and Pattern Recognition (CVPR), 2001*, volume 2, pages 18–24. IEEE Computer Society Press, 2001.

[40] D. Huttenlocher and S. Ullman. Object recognition using alignment. In *Proceedings, First International Conference on Computer Vision*, pages 102–111, London, UK, 1987.

[41] Daniel P. Huttenlocher and Shimon Ullman. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5:2:195–212, 1990.

[42] T. Iijima. Basic theory on normalization of a pattern (in case of typical one-dimensional pattern). *Bulletin of Electrical Laboratory*, 26:368–388, 1962. (in Japanese).

[43] S. Irani and P. Raghavan. Combinatorial and experimental results for randomized point matching algorithms. In *SCG '96: Proceedings of the twelfth annual symposium on Computational geometry*, pages 68–77, New York, NY, USA, 1996. ACM Press.

[44] B. Janssen, F. M. W. Kanters, R. Duits, L. M. J. Florack, and B. M. ter Haar Romeny. A linear image reconstruction framework based on Sobolev type inner products. *International Journal of Computer Vision*, 70(3):231–240, December 2006.

[45] *Jext: Source code editor*. http://www.jext.org/.

[46] F. Kanters, L. Florack, R. Duits, and B. Platel. Scalespaceviz: Visualizing α-scale spaces. Demonstration software presentend at the Eighth European Conference on Computer Vision, Prague, Czech Republic, May 2004.

[47] F. M. W. Kanters, M. Lillholm, R. Duits, B. Janssen, B. Platel, L. M. J. Florack, and B. M. ter Haar Romeny. On image reconstruction from multiscale top points. In Kimmel et al. [53], pages 431–442.

[48] F.M.W Kanters, L.M.J. Florack, B. Platel, and B.M. ter Haar Romeny. Multi-scale differential error: A novel image quality assessment tool. In *Proc. of the 8th Int. conf. on Signal and Image Processing 2006,Honolulu, Hawaii.*, pages 188–194, August 2006.

[49] Frans Kanters, Trip Denton, Ali Shokoufandeh, and Luc Florack. Combining different types of scale space interest points using canonical sets. In *First International Conference on Scale Space Methods and Variational Methods in Computer Vision.*, 2007.

[50] Frans Kanters, Bram Platel, Luc Florack, and Bart M. ter Haar Romeny. Content based image retrieval using multiscale top points a feasibility study. In *Scale Space*

Methods in Computer Vision, volume 2695 of *Lecture Notes in Computer Science*, pages 33–43. Springer-Verlag, August 2003.

[51] Yan Ke and Rahul Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition (CVPR), 2004*, volume 2, pages 506–513. IEEE Computer Society Press, 2004.

[52] Jens Keuchel, Matthias Heiler, and Christoph Schnorr. Hierarchical image segmentation based on semidefinite programming. In *Pattern Recognition (Lecture Notes in Computer Science, Vol 3175)*, pages 120–128. Springer-Verlag, August 2004.

[53] R. Kimmel, N. Sochen, and J. Weickert, editors. *Scale Space and PDE Methods in Computer Vision: Proceedings of the Fifth International Conference, Scale-Space 2005, Hofgeismar, Germany*, volume 3459 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, April 2005.

[54] J. Koenderink and A. van Doorn. The internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32:211–216, 1979.

[55] J. J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[56] J. J. Koenderink. A hitherto unnoticed singularity of scale-space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1222–1224, November 1989.

[57] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *International Conference on Machine Learning*, pages 284–292, 1996.

[58] Jay Kothari, Trip Denton, Spiros Mancoridis, and Ali Shokoufandeh. On computing the canonical features of software systems. In *Proceedings of the 13th Working Conference on Reverse Engineering (WCRE 2006, Benevento, October 23-27),*. IEEE Computer Society, 2006.

[59] Jay Kothari, Trip Denton, Ali Shokoufandeh, Spiros Mancoridis, and Ahmed E. Hassan. Studying the evolution of sofware systems using change clusters. In *Proceedings of the 14th International Conference on Program Comprehension (ICPC 2006, Athens, June 14-16),*. IEEE Computer Society, 2006.

[60] A. Kuijper and L. M. J. Florack. Hierarchical pre-segmentation without prior knowledge. In *Proceedings of the 8th International Conference on Computer Vision (Vancouver, Canada, July 9–12, 2001)*, pages 487–493. IEEE Computer Society Press, 2001.

[61] A. Kuijper and L.M.J. Florack. Understanding and modeling the evolution of critical points under gaussian blurring. In *Proceedings of the 7th European Conference on Computer Vision (ECCV)*, volume LNCS 2350, pages 143–157, 2002.

[62] A. Kuijper and L.M.J. Florack. The hierarchical structure of images. *IEEE-Transactions-on-Image-Processing*, 12(9):1067–1079, Sept. 2003.

[63] P. Langley. Selection of relevant features in machine learning. In *AAAI Fall Symposium on Relevance*, pages 140–144, 1994.

[64] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[65] M. Lillholm, M. Nielsen, and L. D. Griffin. Feature-based image analysis. *International Journal of Computer Vision*, 52(2/3):73–95, 2003.

[66] T. Lindeberg. Scale-space behaviour of local extrema and blobs. *Journal of Mathematical Imaging and Vision*, 1(1):65–99, March 1992.

[67] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):117–156, November 1998.

[68] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, November 1998.

[69] Tony Lindeberg. Detecting Salient Blob–Like Image Structures and Their Scales With a Scale–Space Primal Sketch—A Method for Focus–of–Attention. *IJCV*, 11(3):283–318, December 1993.

[70] Tony Lindeberg and Lars Bretzner. Real-time scale selection in hybrid multi-scale representations. In *Lecture Notes in Computer Science*, volume 2695 of *Lecture Notes in Computer Science*, pages 148–163. Springer-Verlag, January 2003.

[71] H. Liu and H. Motoda. Feature transformation and subset selection. *IEEE Intelligent Systems*, 13(2):26–28, 1998.

[72] H. Liu, H. Motoda, and L. Yu. Feature selection with selective sampling. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 395–402, 2002.

[73] D. G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[74] David Lowe. *Demo Software: SIFT Keypoint Detector*, 2005.

[75] David G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the International Conference on Computer Vision ICCV, Corfu*, pages 1150–1157, 1999.

[76] Aidong Lu, Ross Maciejewski, and David S. Ebert. Volume composition using eye tracking data. In *IEEE-VGTC Symposium on Visualization, (Eurovis2006)*, 2006.

[77] L. Lucchese and S.K. Mitra. Color image segmentation: A state-of-the-art survey. In *Proc. of the Indian National Science Academy (INSA-A)*, 2001. New Delhi, India, Vol. 67, A, No. 2, March 2001, pp. 207-221.

[78] David J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[79] Sanjeev Mahajan and H. Ramesh. Derandomizing approximation algorithms based on semidefinite programming. *SIAM Journal on Computing*, 28(5):1641–1663, 1999.

[80] P. Meer, R. Lenz, and S. Ramakrishna. Efficient invariant representations. *Int. J. Comput. Vision*, 26(2):137–152, 1998.

[81] Kaisa M. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Dordrecht, Netherlands, 1999.

[82] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, October 2004.

[83] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, October 2005.

[84] Krystian Mikolajczyk, Bastian Leibe, and Bernt Schiele. Local features for object class recognition. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 2*, pages 1792–1799, Washington, DC, USA, 2005. IEEE Computer Society.

[85] David Mount. *A testbed for k-means clustering algorithms based on local search*, 2005.

[86] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library: Coil. Technical Report CUCS-005-96, Department of Computer Science, Columbia University, Feb 1996.

[87] M. Nielsen and M. Lillholm. What do features tell about images? In M. Kerckhove, editor, *Scale-Space and Morphology in Computer Vision: Proceedings of the Third International Conference, Scale-Space 2001, Vancouver, Canada*, volume 2106 of *Lecture Notes in Computer Science*, pages 39–50. Springer-Verlag, Berlin, July 2001.

[88] John Novatnack, Trip Denton, Ali Shokoufandeh, and Lars Bretzner. Stable bounded canonical sets and image matching. In *Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, pages 316–331, November 2005.

[89] M. J. Owen. Simple canonical views. In *Proceedings, British Machine Vision Conference (BMVC '05)*, September 2005.

[90] B.-U. Pagel, F. Korn, and C. Faloutsos. Deflating the dimensionality curse using multiple fractal dimensions. In *ICDE '00: Proceedings of the 16th International Conference on Data Engineering*, page 589, Washington, DC, USA, 2000. IEEE Computer Society.

[91] Marcello Pelillo, Kaleem Siddiqi, and Steven W. Zucker. Matching hierarchical structures using association graphs. *Lecture Notes in Computer Science*, 1407, 1998.

[92] Perforce - The Fastest Software Configuration Management System. Available online at `http://www.perforce.com`.

[93] B. Platel, M. Fatih Demirci, A. Shokoufandeh, L. M. J. Florack, F. M. W. Kanters, B. M. ter Haar Romeny, and S. J. Dickinson. Discrete representation of top points via scale space tessellation. In Kimmel et al. [53], pages 73–84.

[94] B. Platel, L. M. J. Florack, F. M. W. Kanters, and E. G. Balmachnova. Using multiscale top points in image matching. In *Proceedings of the 11th International Conference on Image Processing (Singapore, October 24–27, 2004)*, pages 389–392. IEEE, 2004.

[95] A. L. Ratan, W. E. L. Grimson, and III W. M. Wells. Object detection and localization by dynamic template warping. *International Journal on Computer Vision*, 36(2):131–147, 2000.

[96] Marc J. Rochkind. The source code control system. *IEEE Transactions on Software Engineering*, 1(4):364–370, 1975.

[97] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[98] Maher Salah, Trip Denton, Spiros Mancoridis, Ali Shokoufandeh, and Filippos I. Vokolos. Scenariographer: A tool for reverse engineering class usage scenarios from method invocation sequences. In *Proceedings of the 21st International Conference on Software Maintenance (ICSM 2005, Budapest,September 25-30)*, pages 155–164, 2005.

[99] C. Schellewald and C. Schnörr. Subgraph matching with semidefinite programming. In Vito Di Gesù Alberto Del Lungo and Attila Kuba, editors, *Electronic Notes in Discrete Mathematics*, volume 12. Elsevier Science Publishers, 2003.

[100] W. Seales, M. Cutts, C. Yuan, and W. Hu. Object recognition in compressed imagery. *Image and Vision Computing*, 1998.

[101] Claude E Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423,623–656, July,October 1948.

[102] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[103] I. Shimshoni and J. Ponce. Finite resolution aspect graphs of polyhedral objects. In *Proceedings, IEEE Workshop on Qualitative Vision*, pages 140–150, New York, NY, June 1993.

[104] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 30:1–24, 1999.

[105] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings Ninth IEEE International Conference on Computer Vision*, volume 2, pages 1470–1477, October 2003.

[106] B. Smith and L. Rowe. A new family of algorithms for manipulating compressed images. In *IEEE Computer Graphics and Applications*, pages 34–42, Sep 1993.

[107] J. Stewman and K. Bowyer. Creating the perspective projection aspect graph of polyhedral objects. In *Proceedings, IEEE Second International Conference on Computer Vision*, pages 494–500, Tampa, FL, 1988.

[108] Zehang Sun, George Bebis, and Ronald Miller. Object detection using feature subset selection. In *Pattern Recognition*, volume 37, pages 2165–2176. Elsevier, November 2004.

[109] Walter F. Tichy. RCS - a system for version control. *Software - Practice and Experience*, 15(7):637–654, 1985.

[110] N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. In *Proceedings of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

[111] K.C. Toh, M.J. Todd, and R.H. Tutuncu. *SDPT3–a Matlab software package for semidefinite programming*, december 2002.

[112] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin, Germany, second edition, 2003. ISBN 3-540-65367-8.

[113] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13:600–612, April 2004.

[114] D. Weinshall and M. Werman. On view likelihood and stability. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):97–108, February 1997.

[115] A. P. Witkin. Scale-space filtering. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1019–1022, Karlsruhe, Germany, 1983.

[116] Dachuan Xu and Guanghui Liu. Approximation algorithm for max dicut with given sizes of parts. *Acta Mathematicae Applicatae Sinica*, 19(2):289–296, 2003.

[117] Qihui Zhu and Jianbo Shi. Shape from shading: Recognizing the mountains through a global view. In *CVPR 2006*, volume 2, pages 1839–1846, June 2006.

Appendix A. Drexel Object Occlusion Repository (DOOR)

A.1 Overview

The Drexel Object Occlusion Repository is a reference set of images for computer vision and object recognition researchers. The images are constructed by overlapping input objects from the COIL-20 database [86] and occluding them by various amounts. The amount of occlusion for each image is measured at the pixel level. An accompanying text file for each occlusion image describes the input files, occlusion rates etc.

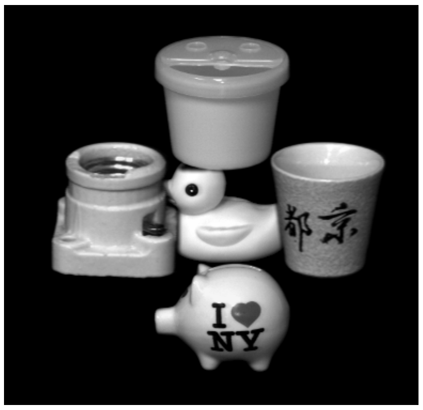


Figure A.1: Occlusion example door_4_41.png

The Drexel Object Occlusion Repository is a syntactic set of images constructed from the COIL-20 database from Columbia University [86]. In a previous work [88] our tests with the COIL-20 database indicated a need for a set of reference scenes where we could test the ability of our algorithm to locate objects in scenes where the objects are partially occluded. In particular we required a precise measurement of the occlusion rates in addition to non-trivial occlusions. To accomplish these goals we have constructed the Drexel

Object Occlusion Repository. The repository contains objects deterministically placed in groupings with varying amounts of occlusion. The repository provides a reference set that researchers in computer vision and object recognition may use for qualitative analysis of algorithmic performance.

The repository is organized into series of images. Series 1 contains images with two objects, series 2 with three objects, series 3 with four objects, series 4 with five objects, and series 6 with five objects scaled and rotated randomly on a background scene. Each series contains 480 images and accompanying text files. The text files include various information about the image including, the name of the objects and percentage of occluded pixels in each of the objects.

A.2 Drexel Object Occlusion Repository (DOOR)

The occluded images consist of a composed set of objects from the COIL-20 database. The objects have not been scaled or rotated, although we plan to expand to the repository in the future to include objects which have been transformed in the image plane. When composing two objects we first create a mask for each image. The background for the image is created by filling edge pixels with red. Then the non-red pixels in the mask are set to green, which designates them as object pixels. The red/green masks for the input image and the occlusion image are then convoluted with a Gaussian. The masks are used to combine the original images. If the pixel is an edge pixel the intensity, I, is given by

$$I = R * R' + G * G',$$

where R, G are the red and green channels of the input image mask and R', G' are the red and green channels of the mask of the image being constructed (occlusion image). This is similar to using an alpha channel for the mixing. For more information on the use of alpha channels consult an appropriate image processing text. After placing an object on the image the amount of occlusion for each object in the image is updated. A pixel is considered occluded if it has been covered by another object, or if it on the border between two objects.

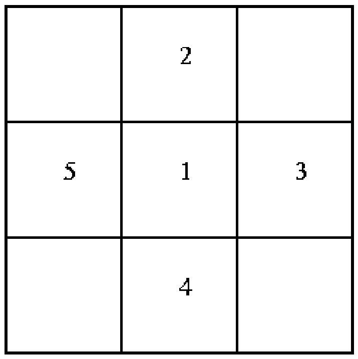


Figure A.2: Object Layout Grid

In general the objects contained in each image are arranged in a grid shown in Figure A.2. For each image an object is placed in the center at position one. Objects are then added at the other positions at varying distances from the center object. As the objects move farther away from the center the amount of occlusion of the center object decreases.

A.3 Contents of the Accompanying Text Files

Each image is accompanied by a text file that provides information about the various objects in the image. The following text file describes image forty-one in series four (shown in Figure A.1).

The file lists the image files of the objects at the various grid positions. Next the file contains the parameters of the Gaussian convolution kernel used to combine the two images at the borders. Many images in the COIL-20 database are cropped such that the object lies

```
Input File1: coil-20-proc/obj1¨0.png
Input File3: coil-20-proc/obj12¨0.png
Input File4: coil-20-proc/obj13¨0.png
Input File5: coil-20-proc/obj14¨0.png
Input File2: coil-20-proc/obj15¨0.png
Output File: door˙4/0-99/door˙4˙41.png
Gaussian˙radius=3
Gaussian˙sigma=2
Border=10
Width=592
Height=592
Object=1 file=coil-20-proc/obj1¨0.png x=222 y=222 occlusion=16.20 oldpixels=6668
  newpixels=5588
Object=2 file=coil-20-proc/obj15¨0.png x=222 y=122 occlusion=0.00 oldpixels=11697
  newpixels=11697
Object=3 file=coil-20-proc/obj12¨0.png x=322 y=222 occlusion=0.00 oldpixels=9975
  newpixels=9975
Object=4 file=coil-20-proc/obj13¨0.png x=222 y=322 occlusion=0.00 oldpixels=7751
  newpixels=7751
Object=5 file=coil-20-proc/obj14¨0.png x=122 y=222 occlusion=0.00 oldpixels=11826
  newpixels=11826
```

on the edge of the image. Prior to composing the objects in the image we first add a border to each image. The size of this border is reported in the text file. Lastly the file contains the position of the $(0, 0)$ pixel of each object in the image, as well as the percent of pixels that have been occluded. In this particular example object one has had 16.2% of its pixels occluded.

A.4 Query Tool

A query tool is provided to assist the researcher in finding occlusion images.

```
./query-door.pl [OPTIONS]
```

```
DESCRIPTION: This program queries a DOOR data file for objects that have
```

certain amounts of occlusion. It prints the names of the files that meet the criteria.

```
OPTIONS
   -help              show this help
   -data=PATH          path to DOOR data file (door˙all.txt)
   -min=NUMBER         minimum occlusion (0.0)
   -max=NUMBER         maximum occlusion (10.0)
   -obj=PATTERN        pattern for object selection (obj*)

Example:

./query-door.pl -min=9.9 -max=10.2 -obj=obj13.*
Finds examples of object 13 that are occluded by 9.9-10.2% using the data
from the file door˙all.txt.

./query-door.pl -min=19.9 -max=20.2
Finds examples of objects that are occluded by 19.9-20.2% using the data
from the file door˙all.txt.
```

A.5 Obtaining the Repository

The repository is on-line at http://aal.cs.drexel.edu/home/door/index.html. A gzipped tar file is available on the website as well.

Appendix B. Notes

B.1 Similarity Measures

In the experiments presented in Chapters 4 through 8, the canonical set method was shown to be useful in a number of scenarios. Since the method depends on having a similarity measure between data points, a natural question is: What is a good similarity measure? Unfortunately, at this point no easy answer to this question is known. To a large extent the proper measure is highly problem dependent. As such, the selection of a similarity measure requires intimate knowledge of the dataset under analysis.

Some general guidelines, however, might be useful. The measure should be informative, that is it should be able to discriminate between as many data points as possible. The reasoning behind this is somewhat obvious. If, the distance between two data points is zero (similarity is unity) then there is no way to tell the difference between them, and the algorithms will not distinguish between them. An example of more informative measures can be seen in Figure 6.19. DSPACE is a more informed measure than XY, as it is the Euclidean distance in descriptor space, rather than the Euclidean distance on the image plane.

Another issue to consider is when selecting a similarity measure is careful thought about what the canonical set method does. It selects representative data points. The notions of what is representative and what is similar are closely tied.

If each data point represents a distribution, the Earth Mover's distance [97] might be a good place to start. If each data point is a graph, then something like the the cardinality of the maximal clique in the computed association graph [91] might work. If each data point is a string, then the edit-distance [64] seems a logical choice.

Conversion between distance measures and similarity measures is relatively straightforward. Common ways are $s = \frac{1}{d}$, $s = \frac{1}{1+d}$, and $s = e^{-d}$, where s is the similarity and d

is the distance.

Vita

Trip Denton was born in Detroit, Michigan in 1956. In 1979 he received his four-year certificate from the Pennsylvania Academy of the Fine Arts in Philadelphia, Pennsylvania. He received his B.S. in General Studies from Drexel University in Philadelphia in 2003, and his M.S. in Computer Science from Drexel University in 2006. Mr. Denton is the recipient of the Graduate Research Award, Drexel University 2007, and the Hill Fellowship, Drexel University 2006. His research interests include algorithm design, computer vision, subset selection and clustering, and nonlinear optimization.

Selected Publications

Trip Denton, Jeff Abrahamson, and Ali Shokoufandeh. Approximation of canonical sets and their application to 2D view simplification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR04)*, volume 2, pages 550-557, June 2004.

Trip Denton, M. Fatih Demirci, Jeff Abrahamson, Ali Shokoufandeh, and Sven Dickinson. Selecting canonical views for view-based 3D object recognition. In *Proceedings of the 17th IAPR International Conference on Pattern Recognition (ICPR04)*, pages 273-276, August 2004.

Trip Denton, John Novatnack, and Ali Shokoufandeh. Drexel Object Occlusion Repository (DOOR). Technical Report DU-CS-05-08, Drexel University, Computer Science Department, 2005.

Frans Kanters, Trip Denton, Ali Shokoufandeh, and Luc Florack. Combining different types of scale space interest points using canonical sets. In *Proceedings of the First International Conference on Scale Space Methods and Variational Methods in Computer Vision*, Ischia, Italy, June 2007.

Jay Kothari, Trip Denton, Spiros Mancoridis, and Ali Shokoufandeh. On computing the canonical features of software systems. In *Proceedings of the 13th Working Conference on Reverse Engineering (WCRE)*, October 2006.

Jay Kothari, Trip Denton, Spiros Mancoridis, Ali Shokoufandeh, and Ahmed E. Hassan. Studying the evolution of software systems using change clusters. In *Proceedings of the International Conference on Program Comprehension (ICPC 2006)*, June 2006.

Jay Kothari, Trip Denton, Ali Shokoufandeh, and Spiros Mancoridis. Reducing program comprehension effort in evolving software by recognizing feature implementation convergence. In *Proceedings of the 15th IEEE Conference on Program Comprehension (ICPC)*, Banff, Canada, June 2007.

John Novatnack, Trip Denton, Ali Shokoufandeh, and Lars Bretzner. Stable bounded canonical sets and image matching. In *Proceedings of the Fifth International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, pages 316-331, November 2005.

Maher Salah, Trip Denton, Spiros Mancoridis, Ali Shokoufandeh, and Filippos I. Vokolos. Scenariographer: A tool for reverse engineering class usage scenarios from method invocation sequences. In *IEEE Proceedings of the 2005 International Conference on Software Maintenance (ICSM'05)*, Budapest, Hungary, September 2005.